
# Gpg4win für Durchblicker

Eine Veröffentlichung des Gpg4win Projekts

Basierend auf einem Original von

Manfred J. Heinze, Karl Bihlmeier, Isabel Kramer,

Dr. Francis Wray und Ute Bahn

Überarbeitet von

Werner Koch

Version 2.0.0 vom 5. April 2006

## Impressum gpg4win



Die Angaben auf der **folgenden Seite** sind nicht mehr korrekt; wir können diese Seite allerdings nicht abändern, da die Regeln der GFDL hier falsch angewandt wurden. Neue Copyright Hinweise sollten deswegen hier eingestellt werden.

## Impressum

Diese Seite darf nicht verändert werden.

Autor: Manfred J. Heinze, TextLab text+media
Beratung: Lutz Zolondz, G-N-U GmbH
Illustrationen: Karl Bihlmeier, Bihlmeier & Kramer GbR
Layout: Isabel Kramer, Bihlmeier & Kramer GbR
Fachtext: Dr. Francis Wray, e-mediate Ltd.
Redaktion: Ute Bahn, TextLab text+media
Auflage, März 2002

# Inhaltsverzeichnis

# 1. Was ist Gpg4win

**Das Projekt Gpg4win (GNU Privacy Guard for Windows) ist eine vom Bundesamt für Sicherheit in der Informationstechnik beauftragte Email-Verschlüsselungssoftware. Gpg4win bezeichnet ein Gesamtpaket, welches die folgenden Programme umfasst:**

**GnuPG**: das Kernstück, die Verschlüsselungs-Software
**GPA**: der GNU Privacy Assistent, eine Schlüsselverwaltung
**WinPT**: Schlüsselverwaltung, unterstützt auch Verschlüsselung per Clipboard
**GPGol**: ein Plugin für Microsoft Outlook, es integriert dort die Bedienung von GnuPG
**GPGee**: ein Plugin für den Windows Explorer, per rechter Maustaste können Dateien verschlüsselt werden
**Sylpheed-Claws**: ein komplettes Email-Programm mit integrierter GnuPG-Bedienung

Mit dem Verschlüsselungsprogramm GnuPG (GNU Privacy Guard) kann jedermann Emails sicher, einfach und kostenlos verschlüsseln. GnuPG kann ohne jede Restriktion privat oder kommerziell benutzt werden. Die von GnuPG eingesetzte Verschlüsselungstechnologie ist sehr sicher und kann nach dem heutigen Stand von Forschung und Technik nicht gebrochen werden.

GnuPG ist Freie Software[1]. Das bedeutet, dass jedermann das Recht hat, sie nach Belieben kommerziell oder privat zu nutzen. Jedermann darf den Quellcode, also die eigentliche Programmierung des Programms, genau untersuchen und auch selbst Änderungen durchführen und diese weitergeben.[2]

Für eine Sicherheits-Software ist diese garantierte Transparenz des Quellcodes eine unverzichtbare Grundlage. Nur so läßt sich die Vertrauenswürdigkeit eines Programmes prüfen.

GnuPG basiert auf dem internationalen Standard OpenPGP (RFC 2440), ist vollständig kompatibel zu PGP und benutzt die gleiche Infrastruktur (Schlüsselserver etc.).

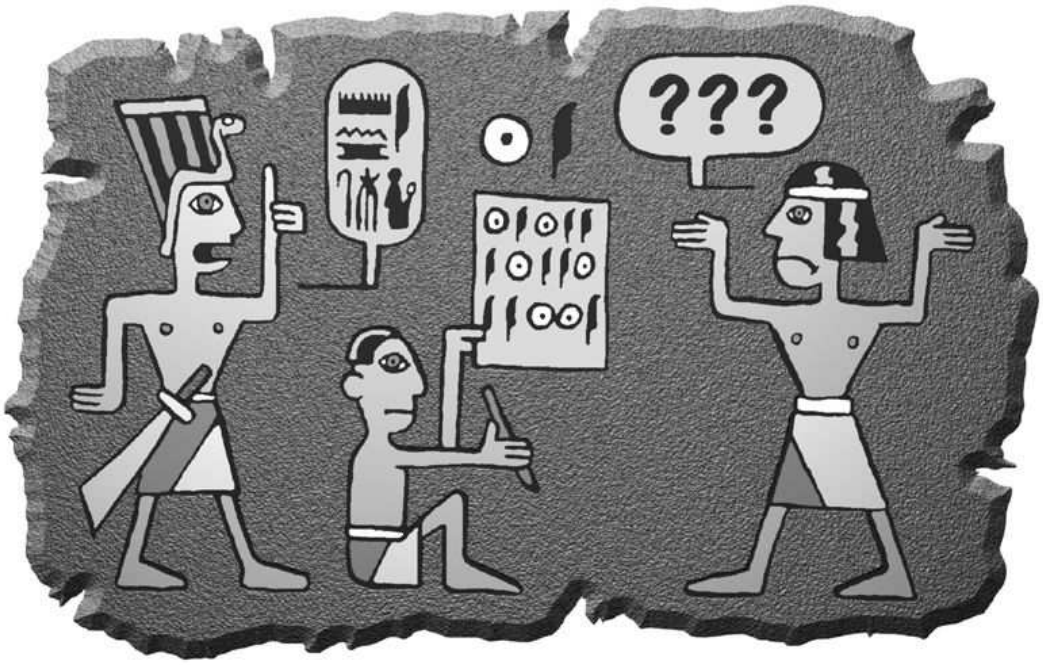
PGP („Pretty Good Privacy“) ist keine Freie Software, sie war lediglich vor vielen Jahren kurzzeitig zu ähnlichen Bedingungen wie GnuPG erhältlich. Diese Version entspricht aber schon lange nicht mehr dem Stand der Technik.

Weitere Informationen zu GnuPG und den Projekten der Bundesregierung zum Schutz des Internets finden Sie auf der Website www.bsi-fuer-buerger.de des Bundesamtes für Sicherheit in der Informationstechnik.

[1] oft ungenau auch als Open Source Software bezeichnet
[2] Obwohl dies ausdrücklich erlaubt ist, sollte man ohne ausreichendes Fachwissen nicht leichtfertig Änderungen durchführen da hierdurch die Sicherheit der Software beeinträchtigt werden kann.

## 2. Warum überhaupt verschlüsseln?

Die Verschlüsselung von Nachrichten wird manchmal als das zweitälteste Gewerbe der Welt bezeichnet. Verschlüsselungstechniken benutzten schon der Pharao Khnumhotep II, Herodot und Cäsar. Dank Gpg4win ist Verschlüsselung nunmehr für jedermann frei und kostenlos zugänglich...




Die Computertechnik hat uns phantastische Mittel in die Hand gegeben, um rund um den Globus miteinander zu kommunizieren und uns zu informieren. Aber Rechte und Freiheiten, die in anderen Kommunikationsformen längst selbstverständlich sind, müssen wir uns in den neuen Technologien erst sichern. Das Internet ist so schnell und massiv über uns hereingebrochen, dass wir mit der Wahrung unserer Rechte noch nicht so recht nachgekommen sind.

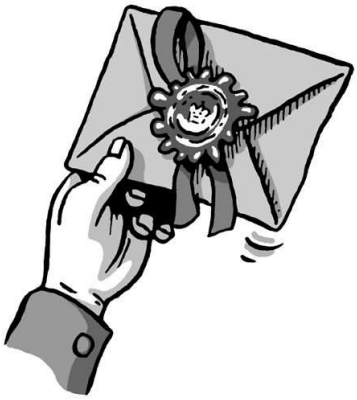
Beim altmodischen Briefschreiben haben wir die Inhalte unserer Mitteilungen ganz selbstverständlich mit einem Briefumschlag geschützt. Der Umschlag schützt die Nachrichten vor fremden Blicken, eine Manipulation am Umschlag kann man leicht bemerken. Nur wenn etwas nicht ganz so wichtig ist, schreibt man es auf eine ungeschützte Postkarte, die auch der Briefträger oder andere lesen können.

Ob die Nachricht wichtig, vertraulich oder geheim ist, das bestimmt man selbst und niemand sonst.

Diese Entscheidungsfreiheit haben wir bei E-Mail nicht. Eine normale E-Mail ist immer offen wie eine Postkarte, und der elektronische „Briefträger“ — und andere — können sie immer lesen. Die Sache ist sogar noch schlimmer: die Computertechnik bietet nicht nur die Möglichkeiten, die vielen Millionen E-Mails täglich zu befördern und zu verteilen, sondern auch, sie zu kontrollieren.

Niemand hätte je ernsthaft daran gedacht, alle Briefe und Postkarten zu sammeln, ihren Inhalt auszuwerten oder Absender und Empfänger zu protokollieren. Das wäre einfach nicht machbar gewesen, oder es hätte zu lange gedauert. Mit der modernen Computertechnik ist das technisch möglich. Es gibt mehr als einen Hinweis darauf, dass dies genau heute schon im großen Stil mit Ihrer und meiner E-Mail geschieht.[3].

Denn: der Umschlag fehlt.



[3]Hier sei nur an das Echelon System erinnert; siehe http://www.heise.de/tp/r4/artikel/6/6928/1.html.

Was wir Ihnen hier vorschlagen, ist ein Umschlag für Ihre elektronischen Briefe. Ob Sie ihn benutzen, wann, für wen und wie oft, ist ganz allein Ihre Sache. Software wie Gpg4win gibt Ihnen lediglich die Wahlfreiheit zurück. Die Wahl, ob Sie persönlich eine Nachricht für wichtig und schützenswert halten oder nicht.

Das ist der Kern des Rechts auf Brief-, Post- und Fernmeldegeheimnis im Grundgesetz, und dieses Recht können Sie mit Hilfe der Software Gpg4win wahrnehmen. Sie müssen sie nicht benutzen — Sie müssen ja auch keinen Briefumschlag benutzen. Aber es ist Ihr gutes Recht.

Um dieses Recht zu sichern, bietet Gpg4win Ihnen sogenannte „starke Verschlüsselungstechnik“. „Stark“ bedeutet hier: mit keinem gegenwärtigen Mittel zu knacken. In vielen Ländern waren starke Verschlüsselungsmethoden bis vor ein paar Jahren den Militärs und Regierungsbehörden vorbehalten. Das Recht, sie für jeden Bürger nutzbar zu machen, haben sich die Internetnutzer mühsam erobert; manchmal auch mit der Hilfe von klugen und weitsichtigen Menschen in Regierungsinstitutionen, wie im Falle der Portierung von GnuPG auf Windows. GnuPG wird von Sicherheitsexperten in aller Welt als eine praktikable und sichere Software angesehen.

Wie wertvoll diese Sicherheit für Sie ist, liegt ganz in Ihrer Hand, denn Sie allein bestimmen das Verhältnis zwischen Bequemlichkeit bei der Verschlüsselung und größtmöglicher Sicherheit. Dazu gehören die wenigen, aber umso wichtigeren Vorkehrungen, die Sie treffen müssen, und die wir im Folgenden besprechen:

# 3. Wie funktioniert Gpg4win?

Das Besondere an Gpg4win und der zugrundeliegenden Public-Key Methode ist, dass sie jeder verstehen kann und soll. Nichts daran ist Geheimwissen – es ist nicht einmal besonders schwer zu verstehen.

Die Benutzung von Gpg4win ist sehr einfach, seine Wirkungsweise dagegen ziemlich kompliziert. Wir werden in diesem Kapitel erklären, wie Gpg4win funktioniert – nicht in allen Details, aber so, dass die Prinzipien dahinter deutlicher werden. Wenn Sie diese Prinzipien kennen, werden Sie ein hohes Vertrauen in die Sicherheit von Gpg4win gewinnen.

Ganz am Ende dieses Buches, in Kapitel 13, können Sie – wenn Sie wollen – auch noch die letzten Geheimnisse um die Public-Key Kryptographie lüften und entdecken, warum Gpg4win nicht zu knacken ist.

### Der Herr der Schlüsselringe

Wenn man etwas sehr Wertvolles sichern will, schließt man es am besten ein — mit einem Schlüssel. Noch besser mit einem Schlüssel, den es nur einmal gibt und den man ganz sicher aufbewahrt.

Denn wenn dieser Schlüssel in die falschen Hände fällt, ist es um die Sicherheit des wertvollen Gutes geschehen. Dessen Sicherheit steht und fällt mit der Sicherheit des Schlüssels. Also hat man den Schlüssel mindestens genauso gut abzusichern, wie das zu sichernde Gut selbst. Die genaue Form des Schlüssels muss völlig geheim gehalten werden.

Geheime Schlüssel sind in der Kryptographie ein alter Hut: schon immer hat man Botschaften geheimzuhalten versucht, indem man den Schlüssel geheimhielt. Dies wirklich sicher zu machen ist sehr umständlich und dazu auch sehr fehleranfällig.

Das Grundproblem bei der „normalen“ geheimen Nachrichtenübermittlung ist, dass für Ver- und Entschlüsselung derselbe Schlüssel benutzt wird und dass sowohl der Absender als auch der Empfänger diesen geheimen Schlüssel kennen.

Dies führt zu einer ziemlich paradoxen Situation: Bevor man mit einem solchen System ein Geheimnis – eine verschlüsselte Nachricht — mitteilen kann, muss man schon vorher ein anderes Geheimnis – den Schlüssel – mitgeteilt haben. Und da liegt der Hase im Pfeffer: man muss sich ständig mit dem Problem herumärgern, dass der Schlüssel unbedingt ausgetauscht werden muss, aber auf keinen Fall von einem Dritten abgefangen werden darf.

Gpg4win dagegen arbeitet – außer mit dem Geheimschlüssel — mit einem weiteren Schlüssel („key“), der vollkommen frei und öffentlich („public“) zugänglich ist.

Man spricht daher auch von Gpg4win als einem „Public-Key“ Verschlüsselungssystem.

Das klingt widersinnig, ist es aber nicht. Der Witz an der Sache: es muss kein Geheimschlüssel mehr ausgetauscht werden. Im Gegenteil: der Geheimschlüssel darf auf keinen Fall ausgetauscht werden! Weitergegeben wird nur der öffentliche Schlüssel – und den kennt sowieso jeder.

Mit Gpg4win benutzen Sie also ein Schlüsselpaar – eine geheime und eine zweite öffentliche Schlüsselhälfte. Beide Hälften sind durch eine komplexe mathematische Formel untrennbar miteinander verbunden. Nach heutiger wissenschaftlicher und technischer Kenntnis ist es unmöglich, einen Schlüsselteil aus dem anderen zu berechnen und damit den Code zu knacken. In Kapitel 13 erklären wir, wie das funktioniert.

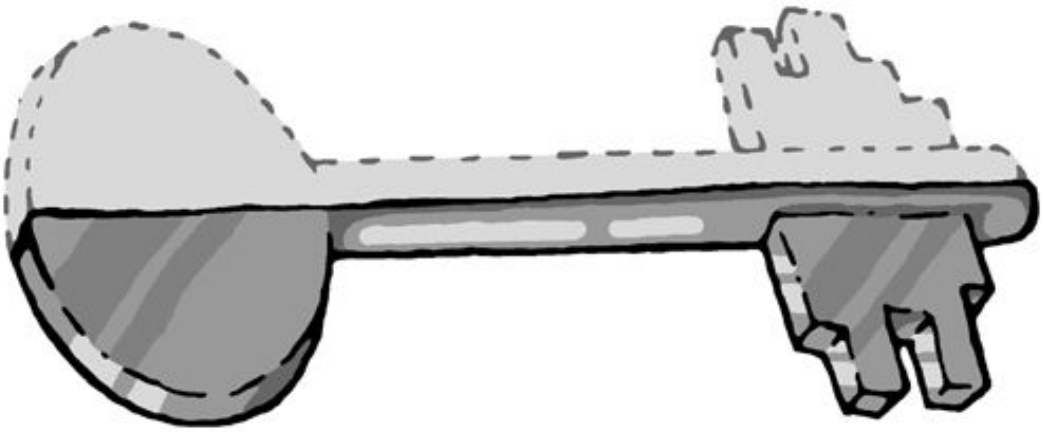
Das Gpg4win-Prinzip ist wie gesagt recht einfach:

Der **geheime Schlüssel**, auch **private Schlüssel** genannt (secret oder private key), muss geheim gehalten werden.

Der **öffentliche Schlüssel** (public key) soll so öffentlich wie möglich gemacht werden.

Beide Schlüsselteile haben ganz und gar unterschiedliche Aufgaben:

der geheime Schlüsselteil **entschlüsselt** Nachrichten



der öffentliche Schlüsselteil **verschlüsselt**.

### Der öffentliche Safe

In einem kleinen Gedankenspiel wird die Methode des Public-Key Verschlüsselungssystems und ihr Unterschied zur „nicht-public-key“ Methode deutlicher:

**Die „nicht-Public-Key Methode“ geht so:**

Stellen Sie sich vor, Sie stellen einen Briefkasten vor Ihrem Haus auf, über den Sie geheime Nachrichten übermitteln wollen.

Der Briefkasten ist mit einem Schloss verschlossen, zu dem es nur einen einzigen Schlüssel gibt. Niemand kann ohne diesen Schlüssel etwas hineinlegen oder herausnehmen. Damit sind Ihre geheimen Nachrichten zunächst einmal gut gesichert.

Da es nur einen Schlüssel gibt, muss Ihr Korrespondenzpartner denselben Schlüssel wie Sie haben, um den Briefkasten damit auf- und zuschließen und eine Geheimnachricht deponieren zu können.

Diesen Schlüssel müssen Sie Ihrem Korrespondenzpartner auf geheimem Wege übergeben.

Erst wenn der andere den Geheimschlüssel hat, kann er den Briefkasten öffnen und die geheime Nachricht lesen.

Alles dreht sich also um diesen Schlüssel: wenn ein Dritter ihn kennt, ist es sofort aus mit den Geheimbotschaften. Sie und Ihr Korrespondenzpartner müssen ihn also genauso geheim austauschen wie die Botschaft selbst.

Aber – eigentlich könnten Sie ihm bei dieser Gelegenheit ja auch gleich die geheime Mitteilung übergeben...

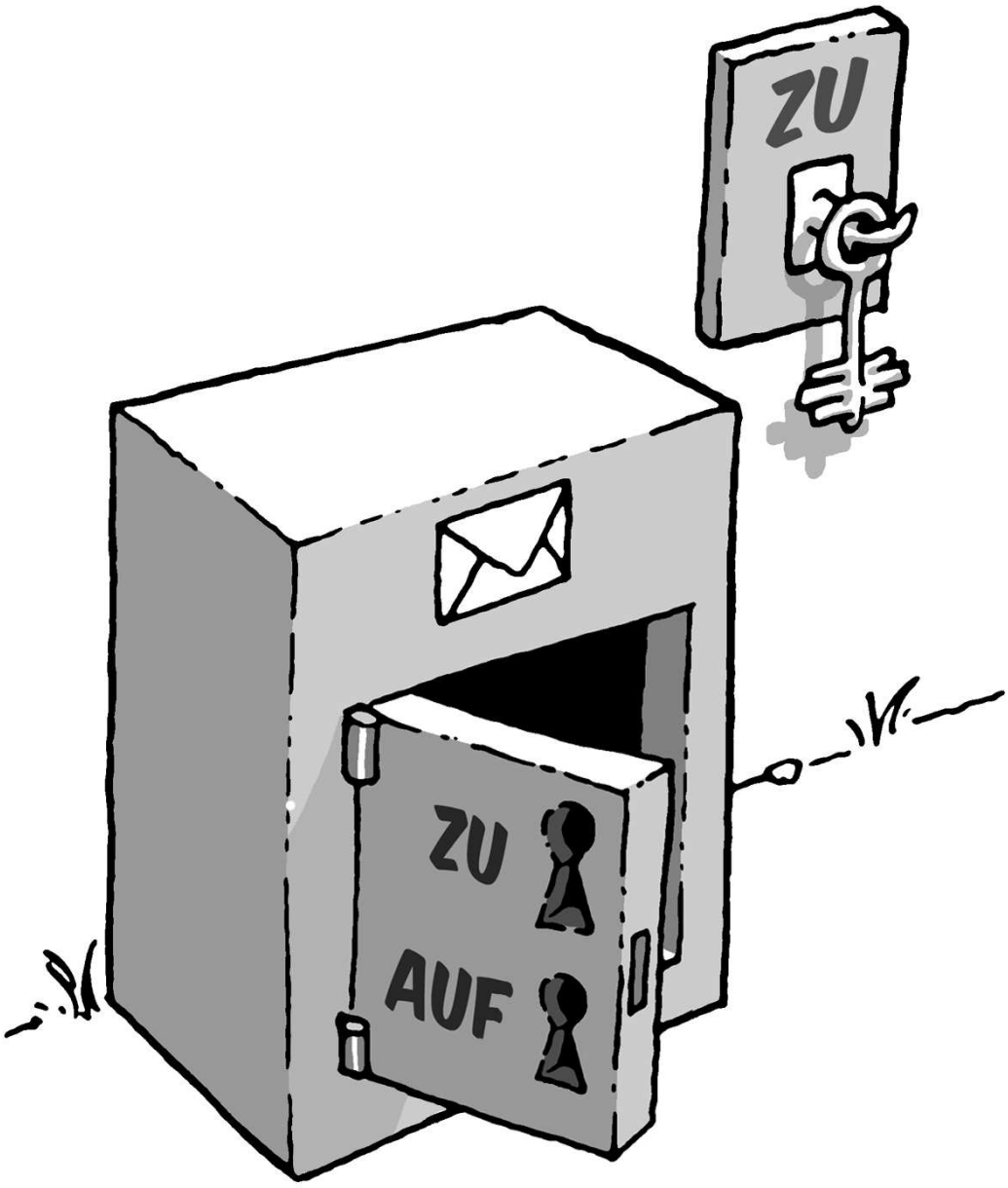
**Übertragen auf die E-Mail-Verschlüsselung:** weltweit müssten alle E-Mailteilnehmer geheime Schlüssel besitzen und auf geheimem Wege austauschen, bevor sie geheime Nachrichten per E-Mail versenden könnten.

Vergessen wir diese Möglichkeit am besten sofort wieder...

**Jetzt die Public-Key Methode:**

Sie installieren wieder einen Briefkasten vor Ihrem Haus. Aber: dieser Briefkasten ist – ganz im Gegensatz zu dem ersten Beispiel — stets offen. Direkt daneben hängt – weithin öffentlich sichtbar — ein Schlüssel, mit dem jedermann den Briefkasten zuschließen kann.

**Zuschließen, aber nicht aufschließen:** das ist der Trick.




Dieser Schlüssel gehört Ihnen, und — Sie ahnen es: es ist Ihr öffentlicher Schlüssel.

Wenn jemand Ihnen eine geheime Nachricht hinterlassen will, legt er sie in den Briefkasten und schließt mit Ihrem öffentlichen Schlüssel ab. Jedermann kann das tun, denn der Schlüssel dazu ist ja völlig frei zugänglich.

Kein anderer kann den Briefkasten nun öffnen und die Nachricht lesen. Selbst derjenige, der die Nachricht in dem Briefkasten eingeschlossen hat, kann ihn nicht wieder aufschließen, zum Beispiel um die Botschaft nachträglich zu verändern.

Denn die öffentliche Schlüsselhälfte taugt ja nur zum Abschließen.

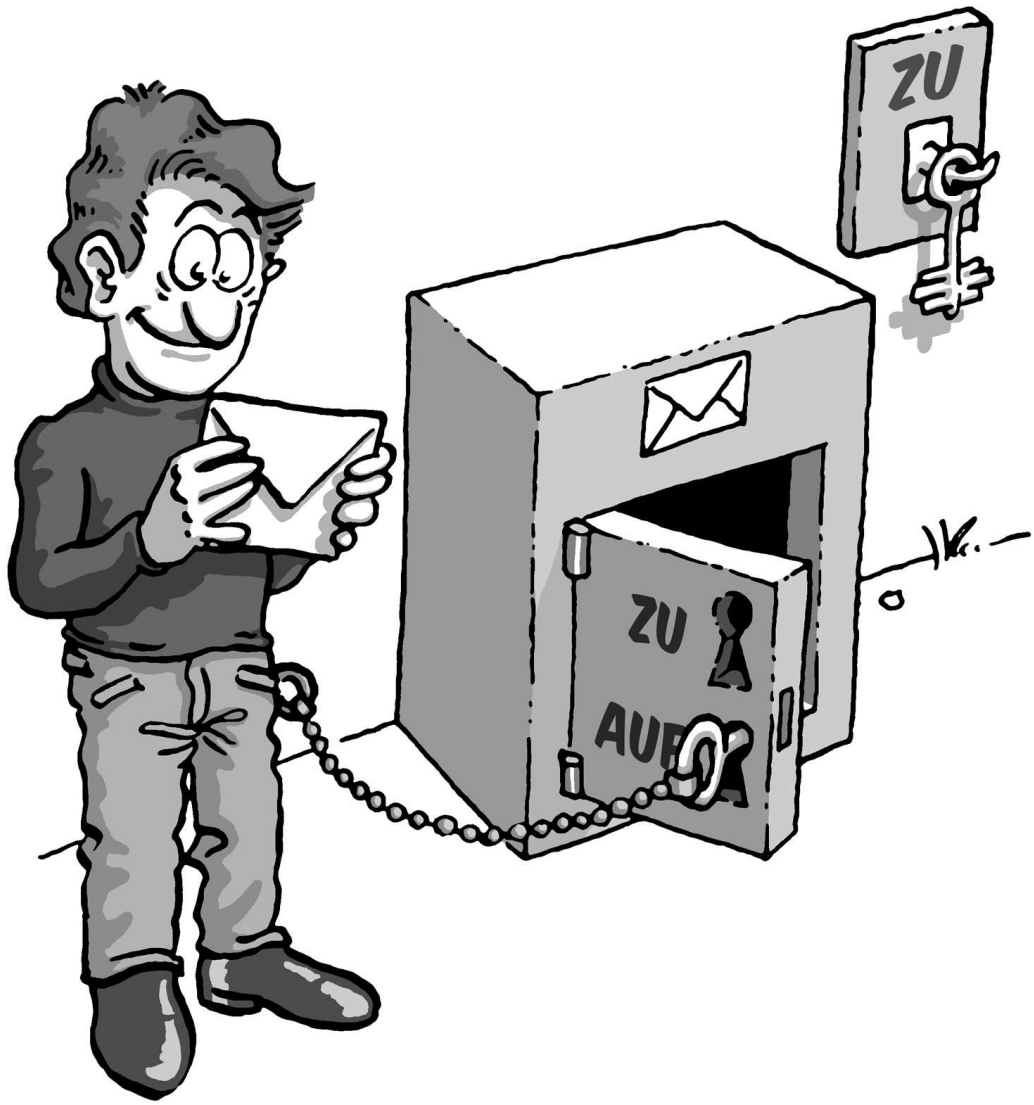
Aufschließen kann man den Briefkasten nur mit einem einzigen Schlüssel: Ihrem eigenen geheimen oder privaten Schlüsselteil.

**Wieder übertragen auf die E-Mail-Verschlüsselung:** jedermann kann eine E-Mail an Sie verschlüsseln. Er benötigt dazu keineswegs einen geheimen, sondern ganz im Gegenteil einen vollkommen öffentlichen, „ungeheimen“ Schlüssel. Nur ein einziger Schlüssel entschlüsselt die E-Mail wieder: Ihr privater, geheimer Schlüssel.

Spielen wir das Gedankenspiel noch einmal anders herum:

Wenn Sie einem anderen eine geheime Nachricht zukommen lassen wollen, benutzen Sie dessen Briefkasten mit seinem öffentlichen, frei verfügbaren Schlüssel.

Sie müssen Ihren Briefpartner dazu nicht persönlich kennen, ihn getroffen oder je mit ihm gesprochen haben, denn sein öffentlicher Schlüssel ist überall und jederzeit zugänglich. Wenn Sie Ihre Nachricht hinterlegt und den Briefkasten des Empfängers mit seinem öffentlichem Schlüssel wieder verschlossen haben, ist sie völlig unzugänglich für jeden anderen, auch für Sie selbst. Nur der Empfänger kann den Briefkasten mit seinem privaten Schlüssel öffnen und die Nachricht lesen.



**Was ist nun eigentlich gewonnen:** es gibt immer noch einen geheimen Schlüssel!?

Der Unterschied gegenüber der „nicht-Public-Key Methode“ ist allerdings ein gewaltiger:

Ihren privater Schlüssel kennen und benutzen nur Sie selbst. Er wird niemals einem Dritten mitgeteilt – die Notwendigkeit einer geheimen Übergabe entfällt, sie verbietet sich sogar.

Es muss überhaupt nichts Geheimes mehr zwischen Absender und Empfänger ausgetauscht werden — weder eine geheime Vereinbarung noch ein geheimes Codewort.

Das ist – im wahrsten Sinne des Wortes — der Knackpunkt: alle „alten“ Verschlüsselungsverfahren können geknackt werden, weil ein Dritter sich beim Schlüsselaustausch in den Besitz des Schlüssels bringen kann.

Dieses Risiko entfällt, weil der Geheimschlüssel nicht ausgetauscht wird und sich nur an einem einzigen Ort befindet: Ihrem eigenen Schlüsselbund.

# 4. Der Passwortsatz

Wie Sie oben gesehen haben, ist der private Schlüssel eine der wichtigsten Komponenten im Public-Key Verschlüsselungssystem. Man muss (und darf) ihn zwar nicht mehr auf geheimem Wege mit seinen Korrespondenzpartnern austauschen, aber nach wie vor ist seine Sicherheit der Schlüssel zur Sicherheit des „ganzen“ Systems.

Es ist deswegen eminent wichtig, diesen private Schlüssel sicher abzuspeichern. Dies geschieht auf zweierlei Weise:

Jeder andere Benutzer des Rechners, auf dessen Festplatte dieser Schlüssel gespeichert ist, darf keinen Zugriff auf ihn erhalten – weder zum schreiben noch zum lesen. Es ist deswegen unbedingt zu vermeiden, den Schlüssel in einem öffentlichen Ordner (z.B. `c:\Temp` oder `c:\WINNT`) abzulegen. Gpg4win speichert den Schlüssel deswegen im sogenannten „Heimverzeichnis“ („Homedir“) von GnuPG ab. Dies kann sich je nach System an unterschiedlichen Orten befinden; für einen Benutzer mit dem Anmeldenamen „Harry“ könnte es z.B.:
`C:\Dokumente und Einstellungen\harry\Anwendungsdaten\gnupg`
sein. Der geheime Schlüssel befindet sich dort in eine Datei mit dem Namen `secring.gpg`.

Dieser Schutz allein ist allerdings nicht ausreichend: Zum einen kann der Administrator des Rechners immer auf alle Dateien zugreifen — also auch auf Ihren geheimen Schlüssel. Zum anderen könnte der Rechner abhanden kommen oder durch „Malware“ (Viren-, Würmer-, Trojanersoftware) kompromittiert werden.

Ein weiterer Schutz ist deswegen notwendig. Dieser besteht aus einem Passwortsatz („passphrase“).

Passwortsatz, weil er aus einem Satz und nicht nur aus einem Wort bestehen soll. Sie müssen diesen Passwortsatz wirklich „im Kopf“ haben und niemals aufschreiben müssen.

Trotzdem darf er nicht erraten werden können. Das klingt vielleicht widersprüchlich, ist es aber nicht. Es gibt einige erprobte Tricks, mit deren Hilfe man sich einen völlig individuellen, leicht zu merkenden und nur sehr schwer zu erratenden Passwortsatz ausdenken kann.

Eine guter Passwortsatz kann so entstehen:

Denken Sie an einen Ihnen gut bekannten Satz, z.B.: Ein blindes Huhn findet auch einmal ein Korn

Aus diesem Satz nehmen Sie zum Beispiel jeden dritten Buchstaben:

```
nieuf dahn lnr
```

Diesen Buchstabensalat kann man sich zunächst nicht unbedingt gut merken, aber man kann ihn eigentlich nie vergessen, solange man den ursprünglichen Satz im Kopf hat. Im Laufe der Zeit und je öfter man ihn benutzt, prägt sich so ein Passwortsatz ins Gedächtnis. Erraten kann ihn niemand.

Denken Sie an ein Ereignis, das sich bereits fest in Ihrem persönlichen Langzeitgedächtnis verankert hat. Vielleicht gibt es einen Satz, mit dem sich Ihr Kind oder Ihr Partner „unvergesslich“ gemacht hat. Oder eine Ferienerinnerung, oder der Titel eines für Sie wichtigen Liedes.

Verwenden Sie kleine und große Buchstaben, Nummern, Sonder- und Leerzeichen durcheinander. Im Prinzip ist alles erlaubt, auch „Ö“, „ß“, „$“ usw.

Aber Vorsicht — falls Sie Ihren geheimen Schlüssel im Ausland an einem fremden Rechner benutzen wollen, bedenken Sie, dass fremdsprachige Tastaturen diese Sonderzeichen oft nicht haben. Beispielsweise werden Sie kein „ä“ auf einer englischen Tastatur finden.

Machen Sie Rechtschreibfehler, z.B. „feLer“ statt „Fehler“. Natürlich müssen Sie sich diese „feLer“ gut merken können. Oder wechseln Sie mittendrin die Sprache. Aus dem schönen Satz

In München steht ein Hofbräuhaus

könnten man beispielsweise diesen Passwortsatz machen:

```
inMinschen stet 1h0f breuhome
```

denken Sie sich einen Satz aus, der möglichst unsinnig ist, den Sie sich aber doch merken können, wie z.B.:

Es blaut so garstig beim Walfang, neben Taschengeld, auch im Winter.

Ein Passwortsatz in dieser Länge ist ein sicherer Schutz für den geheimen Schlüssel.

Es darf auch kürzer sein, wenn Sie einige Buchstaben groß schreiben, z.B. so:

Es blAut nEBen TaschengeLd auch im WiNter.

Kürzer, aber nicht mehr so leicht merken. Wenn Sie einen noch kürzeren Passwortsatz verwenden, indem Sie hier und da Sonderzeichen benutzen, haben Sie zwar bei der Eingabe weniger zu tippen, aber die Wahrscheinlichkeit, dass Sie Ihr Passwortsatz vergessen, wird dabei noch größer.

Ein extremes Beispiel für einen möglichst kurzen, aber dennoch sehr sicheren Passwortsatz ist dieses hier:

```
R!Qw"s,UIb *7\$
```

In der Praxis haben sich solche Zeichenfolgen allerdings als recht wenig brauchbar herausgestellt, da man einfach zu wenig Anhaltspunkte für die Erinnerung hat.

Ein schlechter Passwortsatz ist blitzschnell geknackt, wenn er:

- schon für einen anderen Zweck benutzt wird; z.B. für einen E-Mail-Account oder Ihr Handy
- aus einem Wörterbuch stammt. Cracker lassen in Minutenschnelle komplette Wörterbücher elektronisch über ein Passwort laufen.
- aus einem Geburtsdatum oder einem Namen besteht. Wer sich die Mühe macht, Ihre E-Mail zu entziffern, kann auch ganz leicht an diese Daten herankommen.
- ein landläufiges Zitat ist wie „das wird böse enden“ oder „to be or not to be“. Auch mit derartigen gängigen Zitaten testen Cracker routinemäßig und blitzschnell ein Passwort.
- aus nur einem Wort oder aus weniger als 8 Zeichen besteht. Denken Sie sich einen längeren Passwortsatz aus, kein Passwort.

Wenn Sie nun Ihren Passwortsatz zusammenstellen, nehmen Sie auf gar keinen Fall eines der oben angeführten Beispiele. Denn es liegt auf der Hand, dass jemand, der sich ernsthaft darum bemüht, Ihr Passwortsatz herauszubekommen, zuerst ausprobieren würde, ob Sie nicht eines dieser Beispiele genommen haben, falls er auch diese Informationen gelesen hat.

Seien Sie kreativ. Denken Sie sich jetzt einen Passwortsatz aus. Unvergesslich und unknackbar.

Lesen Sie dann im Handbuch „Gpg4win für Einsteiger“, Kapitel 4 („Sie erzeugen Ihre Schlüsselpaar“) weiter.

# 5. Schlüssel im Detail

Der Schlüssel, den Sie erzeugt haben, besitzt einige Kennzeichen:

- die Benutzerkennung
- die Schlüsselkennung
- das Verfallsdatum
- das Benutzervertrauen
- das Schlüsselvertrauen

**Die Benutzerkennung** besteht aus dem Namen und der E-Mail-Adresse, die Sie während der Schlüsselerzeugung eingegeben haben, also z.B.
`-Heinrich Heine <heinrichh@gpg4win.de>`.

**Die Schlüsselkennung** verwendet die Software intern um mehrere Schlüssel voneinander zu unterscheiden. Mit dieser Kennung kann man auch nach öffentlichen Schlüsseln suchen, die auf den Keyservern liegen. Was Keyserver sind, erfahren Sie im folgenden Kapitel.

**Das Verfallsdatum** ist normalerweise auf „kein Verfallsdatum“ gesetzt. Sie können das ändern, indem Sie auf die Schaltfläche „Ändern“ klicken und ein neues Ablaufdatum eintragen. Damit können Sie Schlüssel nur für eine begrenzte Zeit gültig erklären, zum Beispiel, um sie an externe Mitarbeiter auszugeben.

**Das Benutzervertrauen** beschreibt das Maß an Zuversicht, das Sie subjektiv in den Besitzer des Schlüssel setzen, andere Schlüssel korrekt zu signieren. Es kann über die Schaltfläche „Ändern“ editiert werden.

**Das Schlüsselvertrauen** schließlich bezeichnet das Vertrauen, das man gegenüber dem Schlüssels hat. Wenn man sich von der Echtheit eines Schlüssels überzeugt und ihn dann auch signiert hat, erhält er volles „Schlüsselvertrauen“.

Diese Angaben sind für die tagtägliche Benutzung des Programms nicht unbedingt wichtig. Sie werden relevant, wenn Sie neue Schlüssel erhalten oder ändern. Wir besprechen die Punkte „Benutzervertrauen“ und „Schlüsselvertrauen“ in Kapitel 9.

# 6. Die Schlüsselserver

Um verschlüsselt mit anderen zu kommunizieren, müssen die Partner ihre Schlüssel veröffentlichen und austauschen. Dazu ist — Sie erinnern sich an Kapitel 1 — keine Geheimniskrämerei notwendig, denn Ihr öffentlicher Schlüsselteil ist ja ganz und gar „ungeheim“.

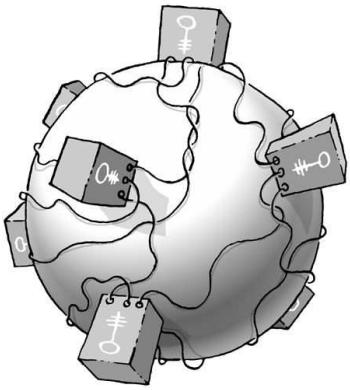
Im Internetzeitalter ist eine möglichst große Verbreitung Ihres öffentlichen Schlüssels überhaupt kein Problem. Sie können ihn z.B. über internationale Keyserver oder per E-Mail publizieren — diese beiden Möglichkeiten haben wir Ihnen im „Einsteiger-Handbuch“ vorgeschlagen. Es gibt aber noch andere:

- Verbreitung des Schlüssels über die eigene Homepage
- als Dateianhang an einer E-Mail
- last but not least: persönlich per USB-Stick oder Diskette

Am praktischsten ist sicher die Veröffentlichung über die Keyserver, die von allen Programmen nach dem OpenPGP-Standard benutzt werden können. Diese Möglichkeit haben wir bereits im Handbuch „Gpg4win für Einsteiger“ Kapitel 6 („Sie veröffentlichen Ihren Schlüssel per Keyserver“) vorgestellt. Es genügt, den Schlüssel an irgendeinen der Keyserver zu senden, denn fast alle synchronsieren sich weltweit miteinander.

Vorsicht: Obwohl es noch keine Hinweise gibt, dass Spammer Adressen wirklich von den Keyservern sammeln, so ist dies jedoch technisch möglich. Falls Sie keinen wirksamen Spamfilter benutzen, sollten Sie u.U. von der Veröffentlichung Ihres Schlüssels auf einem Keyserver absehen.

Ein Keyserver ist in Gpg4win stets voreingestellt. Ein Mausklick genügt, und Ihr Schlüssel ist unterwegs rund um die Welt. Es kann ein, zwei Tage dauern, bis er wirklich überall verfügbar ist, aber dann haben Sie einen globalen Schlüssel! Die Schlüsselserver sind dezentral organisiert, aktuelle Statistiken über ihre Zahl oder die Anzahl der dort liegenden Schlüssel gibt es nicht.



Dieses verteilte Netz von Keyservern sorgt für eine bessere Verfügbarkeit und verhindert dass einzelne Systemandministratoren Schlüssel löschen um so die Kommunikation unmöglich zu machen („Denial of Service“-Angriff).

Wir raten dazu, nur moderne Keyserver zu verwendet (auf denen die SKS Software läuft), da nur diese mit den neueren Merkmalen von OpenPGP umgehen können.

Hier eine Auswahl von gut funktionierenden Keyservern:

- hkp://blackhole.pca.dfn.de
- hkp://pks.gpg.cz
- hkp://pgp.cns.ualberta.ca
- hkp://minsky.surfnet.nl
- hkp://keyserver.ubuntu.com
- hkp://keyserver.pramberger.at
- http://gpg-keyserver.de

- http://keyserver.pramberger.at

Sollte Sie Probleme mit einer Firewall haben, so versuchen Sie am besten die Keyserver, deren Namen mit `http://` beginnen.

Die Keyserver unter den Adressen

- hkp://random.sks.keyserver.penguin.de
- hkp://subkeys.pgp.net

sind ein Sammelpunkt für ein ganzes Netz dieser Server, es wird dann zufällig ein konkreter Server ausgewählt.

Achtung: Der Keyserver `ldap://keyserver.pgp.com` synchronisiert sich nicht mit den anderen Servern und sollte i.d.R. nicht benutzt werden.

Genauso einfach wie Sie einen Schlüssel hochladen, können Sie auf den Keyservern nach einem öffentlichen Schlüssel suchen. Geben Sie in das Suchfeld den Namen des Schlüsselbesitzers ein oder seine E-Mail-Adresse. Als Ergebnis sehen Sie etwa eine solche Ausgabe:

pub 1024/1CE0C630 2006/01/01 ... usw.

und evtl. noch

sig 1CE0C630 ... usw. sig 5B0358A2 ... usw.

Alle drei Eintragungen sind Bestandteil des Schlüssels.

Sie benötigen aber nur den ersten Teil: das ist der öffentliche Schlüssel. Der zweite Teil ist die sogenannte Selbstzertifizierung, der dritte eine Bestätigung der Identität des Schlüsselinhabers.

Klicken Sie nun den Link des ersten Schlüsselteils (pub usw.) an:

Sie sehen den Ihnen schon bekannten Textblock, der den eigentlichen öffentlichen Schlüssel bildet.

Im „Schnelleinstieg“, Kapitel 7 („Sie entschlüsseln eine E-Mail“) und 8 („Sie befestigen einen Schlüssel am Schlüsselbund“) zeigen wir Ihnen, wie man diesen Schlüssel importiert, d.h. am eigenen Gpg4win Schlüsselbund befestigt, und damit eine E-Mail an den Besitzer verschlüsselt.

Diese Suche nach einem Schlüssel funktioniert auch direkt aus Gpg4win: Sie können einfach die E-Mail-Adresse des Schlüsselbesitzers eingeben, oder auch die Schlüsselkennung, falls Ihnen diese bekannt ist. Klicken Sie dazu auf „Import“, und dort auf „Schlüssel vom Key-Server empfangen“.

Gpg4win sucht dann den Schlüssel, importiert ihn und zeigt ihn im Schlüsselverwaltungs-Fenster an.

# 7. Der Schlüssel als Dateianhang

Im Einsteiger Handbuch Kapitel 5 („Sie veröffentlichen Ihren Schlüssel per E-Mail“) haben Sie gesehen, wie einfach man seinen öffentlichen Schlüssel per E-Mail verschicken kann. Wir haben dabei den Schlüssel in einen Ordner exportiert, geöffnet und in die Zwischenablage kopiert. Von dort aus wurde der Schlüssel in ein E-Mail-Programm kopiert und schließlich versandt. Noch einfacher geht es, wenn man den Schlüssel – genau wie im vorherigen Beispiel – exportiert und dann direkt als E-Mail-Anhang verschickt.

Dazu klicken Sie auf im GNU Privacy Assistant auf [Export] in der Iconleiste und dann in dem sich öffnenden Dialog auf [Exportieren in Datei]. Wählen Sie mit [Durchsuchen...] einen geeigneten Ordner auf Ihrem PC, z.B. `C:\Eigene Dateien\` und speichern Sie den Schlüssel dort z.B. als `mein-key.asc`.

Nun ziehen Sie den exportierten Schlüssel als Dateianhang in das entsprechende Fenster Ihres E-Mailprogramms, genauso wie jede andere Datei, und senden sie ihn an den Empfänger.

# 8. PlugIns für E-Mail-Programme

Im „Einsteiger-Handbuch“ haben wir im Kapitel 7 („Sie entschlüsseln eine E-Mail“) erwähnt, dass es PlugIns für bestimmte E-Mail-Programme gibt, die die Ver- und Entschlüsselung erleichtern. Die im Schnelleinstieg vorgestellte Methode mit dem Frontend WinPT funktioniert einfach und schnell, und zwar mit jedem beliebigen E-Mail- und Text-Programm. Trotzdem ist für viele E-Mail-Anwender ein spezieller Programmzusatz in ihrem Lieblings-E-Mailer ein Vorteil.

Plugins für GnuPG gibt es im Moment für folgende Windows-Mailprogramme:

**Thunderbird** mit Plugin **Enigmail**,

**Outlook 2003** mit Plugin **GPGol**, welches in Gpg4win enthalten ist. Läuft nur unter Windows; Outlook sollte nur dann verwendet werden wenn andere organisatorische Vorgaben es bedingen.

**Sylpheed** oder Sylpheed-Claws wie in Gpg4win enthalten. Hier sind im Konfigurationsmenü die Plugins für „PGP/Mime“ und „PGP inline“ zu laden, bei einer Installation über Gpg4win ist das bereits geschehen.

Desweiteren verfügen praktisch alle Mailprogramme, die unter GNU/Linux oder anderen Unix Varianten laufen, über komfortablen und integrierten GnuPG Support.

Da sämtliche Komponenten des Gpg4win Pakets als Freie Software entstehen, ist die Entwicklung stark im Fluss.

Aktuelle Informationen über die Komponenten finden Sie unter www.gpg4win.de.

Informationen zu den Themen IT-Sicherheit, Gpg4win, GnuPG und anderer Software finden Sie auf der Website www.bsi-fuer-buerger.de und www.bsi.de des Bundesamtes für Sicherheit in der Informationstechnik.

# 9. Die Schlüsselprüfung

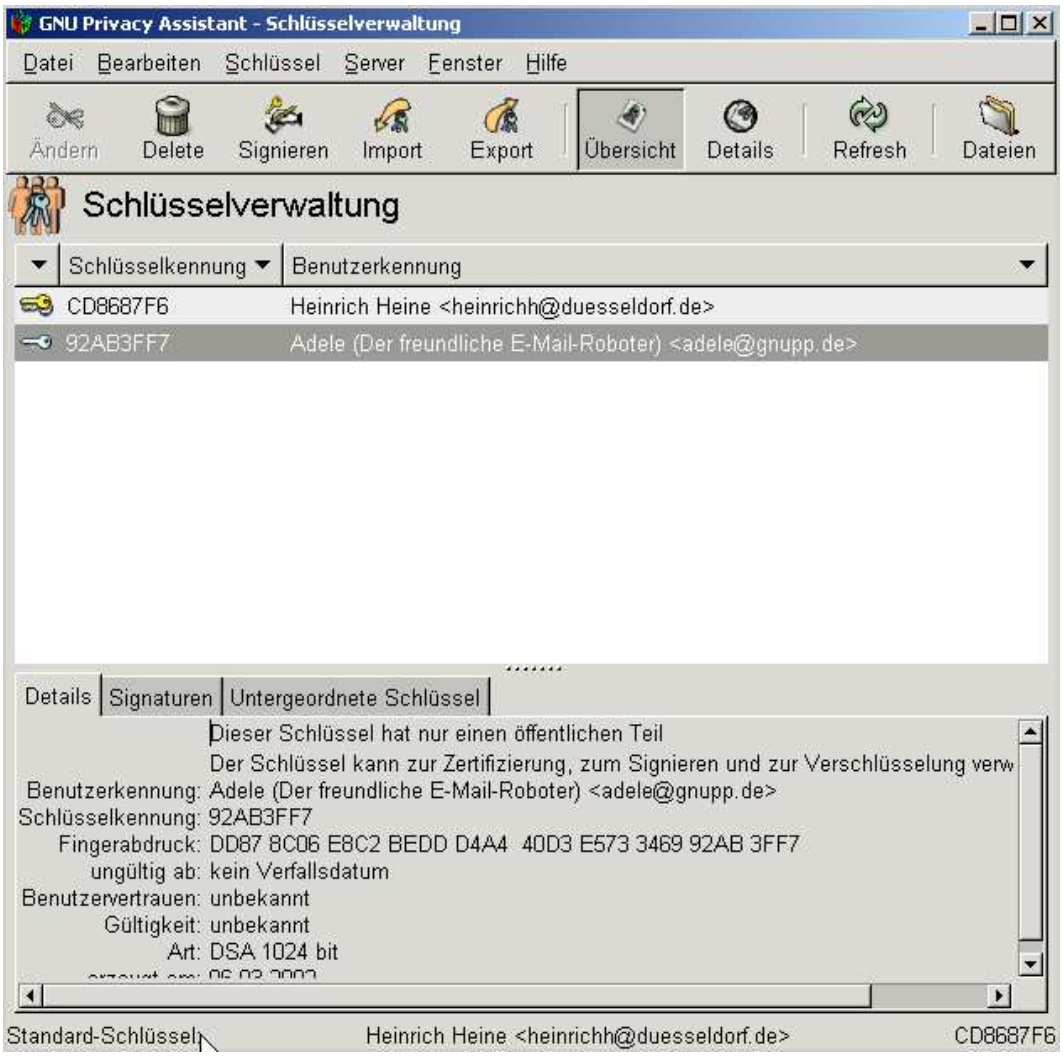
Woher wissen Sie eigentlich, dass der fremde öffentliche Schlüssel wirklich vom Absender stammt? Und umgekehrt — warum sollte Ihr Korrespondenzpartner glauben, dass der öffentliche Schlüssel, den Sie ihm geschickt haben, auch wirklich von Ihnen stammt? Die Absenderangabe auf einer E-Mail besagt eigentlich gar nichts.

Wenn Ihre Bank z.B. eine E-Mail mit Ihrem Namen und der Anweisung erhält, Ihre sämtliche Guthaben auf ein Nummernkonto auf den Bahamas zu überweisen, wird sie sich hoffentlich weigern — E-Mail-Adresse hin oder her. Eine E-Mail-Adresse besagt überhaupt nichts über die Identität des Absenders.

Wenn Sie nur einen kleinen Kreis von Korrespondenzpartnern haben, ist die Sache mit der Identität schnell geregelt: Sie prüfen den Fingerabdruck des anderen Schlüssels.

Jeder öffentliche Schlüssel trägt eine einmalige Kennzeichnung, die ihn zweifelsfrei identifiziert; besser noch als ein Fingerabdruck einen Menschen. Deshalb bezeichnet man diese Kennzeichnung eben als „Fingerprint“.

Wenn Sie einen Schlüssel im GNU Privacy Assistant anklicken, sehen Sie im unteren Teil des Fensters u.a. den Fingerprint:




Der Fingerprint von Adeles Schlüssel ist also:

```
DD87 8C06 E8C2 BEDD D4A4  40D3 E573 3469 92AB 3FF7
```

Wie gesagt — der Fingerprint identifiziert den Schlüssel und seinen Besitzer eindeutig.

Rufen Sie Ihren Korrespondenzpartner einfach an, und lassen Sie sich von ihm den Fingerprint seines Schlüssels vorlesen. Wenn die Angaben mit dem Ihnen vorliegenden Schlüssel übereinstimmen, haben Sie eindeutig den richtigen Schlüssel.

Natürlich können Sie sich auch persönlich mit dem Eigentümer des Schlüssels treffen oder auf jedem anderen Wege mit ihm kommunizieren, solange Sie ganz sicher sind, dass Schlüssel und Eigentümer zusammen gehören. Häufig ist der Fingerprint auch auf Visitenkarten abgedruckt; wenn Sie also eine authentische Visitenkarte haben, so können Sie sich den Anruf ersparen.

Nachdem Sie sich „per Fingerabdruck“ von der Echtheit des öffentlichen Schlüssel überzeugt haben, sollten Sie ihn signieren. Damit teilen Sie anderen Gpg4win-Benutzern mit, dass Sie diesen Schlüssel für echt halten: Sie übernehmen so etwas wie die „Patenschaft“ über diesen Schlüssel und erhöhen das allgemeine Vertrauen in seine Echtheit.

Klicken Sie dazu den betreffenden Schlüssel an und wählen Sie dann „Signieren“ aus der GPA-Menüleiste. Klicken Sie im nun folgenden Hinweis nur dann auf [ Ja ], wenn Sie hundertprozentig sicher sind, den richtigen Schlüssel zu signieren.

Geben Sie nun Ihren Passwortsatz ein und klicken Sie auf [ OK ]. Damit haben Sie mit Ihrem geheimen Schlüssel die Echtheit des Schlüssels bestätigt.

Da — wie Sie wissen — geheimer und öffentlicher Schlüssel untrennbar zusammengehören, kann jedermann mit Hilfe Ihres öffentlichen Schlüssels überprüfen, dass diese Signatur von Ihnen stammt und dass der Schlüssel nicht verändert wurde, also authentisch ist. Damit ist für einen Dritten — wenn auch indirekt — ein gewisses Vertrauen in die Echtheit und Gültigkeit des signierten Schlüssels gegeben.

### Das Netz des Vertrauens

So entsteht — auch über den Kreis von Gpg4win-Benutzern Ihrer täglichen Korrespondenz hinaus — ein „Netz des Vertrauens“, bei dem Sie nicht mehr zwangsläufig darauf angewiesen sind, einen Schlüssel direkt zu prüfen.


Natürlich steigt das Vertrauen in die Gültigkeit eines Schlüssels, wenn mehrere Leute ihn signieren. Ihr eigener öffentlicher Schlüssel wird im Laufe der Zeit die Signatur vieler anderer GnuPG-Benutzer tragen. Damit können immer mehr Menschen darauf vertrauen, dass dieser öffentliche Schlüssel wirklich Ihnen und niemandem sonst gehört.

Wenn man dieses „Web of Trust“ weiterspinnt, entsteht eine flexible Beglaubigungs-Infrastruktur.

Eine einzige Möglichkeit ist denkbar, mit dem man diese Schlüsselprüfung aushebeln kann: jemand schiebt Ihnen einen falschen öffentlichen Schlüssel unter. Also einen Schlüssel, der vorgibt, von X zu stammen, in Wirklichkeit aber von Y ausgetauscht wurde. Wenn ein solcher gefälschter Schlüssel signiert wird, hat das „Netz des Vertrauens“ natürlich ein Loch. Deshalb ist es so wichtig, sich zu vergewissern, ob ein öffentlicher Schlüssel, wirklich zu der Person gehört, der er zu gehören vorgibt.

Was aber, wenn eine Bank oder Behörde überprüfen möchte, ob die Schlüssel ihrer Kunden echt sind? Alle anzurufen, kann hier sicher nicht die Lösung sein...

### Zertifizierungsinstanzen

Hier braucht man eine „übergeordnete“ Instanz, der alle Benutzer vertrauen können. Sie überprüfen ja auch nicht persönlich den Personalausweis eines Unbekannten durch einen Anruf beim Einwohnermeldeamt, sondern vertrauen darauf, dass die ausstellende Behörde diese Überprüfung korrekt durchgeführt und beglaubigt hat.

Solche Zertifizierungsinstanzen gibt es auch bei der Public-Key Verschlüsselung. In Deutschland bietet unter anderem z.B. die Zeitschrift c't schon lange einen solchen Dienst kostenlos an, ebenso wie viele Universitäten.

Wenn man also einen öffentlichen Schlüssel erhält, dem eine Zertifizierungsstelle per Signatur seine Echtheit bestätigt, kann man sich darauf verlassen.

Derartige Beglaubigungsinstanzen oder „Trust Center“ sind auch bei anderen Verschlüsselungssystemen vorgesehen, allerdings sind sie hierarchisch strukturiert: es gibt eine „Oberste Beglaubigungsinstanz“, die „Unterinstanzen“ mit dem Recht zur Beglaubigung besitzt.

Am besten ist diese Infrastruktur mit einem Siegel vergleichbar: die Plakette auf Ihrem Autonummernschild kann Ihnen nur eine dazu berichtigte Institution geben, die die Befugnis dazu wiederum von einer übergeordneten Stelle erhalten hat.

Mit der hierarchischen Zertifizierungs-Infrastruktur entspricht dieses Modell natürlich wesentlich besser den Bedürfnissen staatlicher und behördlicher Instanzen als das lose, auf gegenseitigem Vertrauen beruhende „Web of Trust“ der GnuPG- und PGP-Modelle. Der Kern der Beglaubigung selbst ist allerdings völlig identisch: wenn man in Gpg4win zusätzlich eine hierarchische Zertifizierungsstruktur einbauen würde, dann würde auch Gpg4win dem strengen Signaturgesetz der Bundesrepublik entsprechen.

Wenn Sie sich weiter für dieses Thema interessieren (das zum Zeitpunkt der Arbeit an dieser Gpg4win-Ausgabe gerade in Bewegung ist), dann können Sie sich an der Quelle informieren: die Website „Sicherheit im Internet“ des Bundesministeriums für Wirtschaft und Technologie (www.sicherheit-im-internet.de) hält Sie über dieses und viele andere Themen aktuell auf dem Laufenden.

Eine weitere exzellente, mehr technische Informationsquelle zum Thema der Beglaubigungsinfrastrukturen bietet das Original GnuPG Handbuch, das Sie ebenfalls im Internet finden (www.gnupg.org/gph/de/manual).

# 10. E-Mails signieren

Ganz am Anfang dieses Handbuchs haben wir die E-Mail-Kommunikation mit dem Versenden einer Postkarte verglichen. Erst die Verschlüsselung macht daraus einen Brief mit verschlossenem Umschlag, den nicht mehr jedermann lesen kann.

Gpg4win bietet zusätzlich zur kompletten Verschlüsselung einer E-Mail noch eine weitere Möglichkeit:

- man kann seine E-Mail signieren, mit anderen Worten die E-Mail mit einer elektronischen Unterschrift versehen. Der Text ist dann zwar noch für jeden lesbar, aber der Empfänger kann sicher sein, dass die E-Mail unterwegs nicht manipuliert oder verändert wurde.

Außerdem garantiert die Signatur dem Empfänger, dass die Nachricht auch tatsächlich vom Absender stammt. Und: wenn man mit jemandem korrespondiert, dessen öffentlichen Schlüssel man – aus welchem Grund auch immer — nicht hat, kann man so die Nachricht wenigstens mit dem eigenen privaten Schlüssel „versiegeln“.

Verwechseln Sie diese elektronische Signatur nicht mit den E-Mail-„Signaturen“, die man unter eine E-Mail setzt und die zum Beispiel Ihre Telefonnummer, Ihre Adresse und Ihre Webseite enthalten.

Während diese E-Mail-Signaturen einfach nur als eine Art Visitenkarte fungieren, schützt die elektronische Signatur Ihre E-Mail vor Manipulationen und bestätigt den Absender.

Übrigens ist diese elektronische Unterschrift auch nicht mit der qualifizierten digitalen Signatur gleichzusetzen, wie sie im Signaturgesetz vom 22. Mai 2001 in Kraft getreten ist. Für die private oder berufliche E-Mail-Kommunikation erfüllt sie allerdings genau denselben Zweck.

## 10.1. Signieren mit dem Geheimschlüssel

Tatsächlich ist die Signierung einer E-Mail noch einfacher als die Verschlüsselung: Wie im „Einsteiger-Handbuch“ im Kapitel 9, „Sie verschlüsseln eine E-Mail“, besprochen, schreiben Sie Ihre Nachricht und kopieren sie mit dem Menübefehl „Kopieren“ oder mit dem Tastaturkürzel Strg+C in die Zwischenablage (Clipboard) Ihres Rechners.

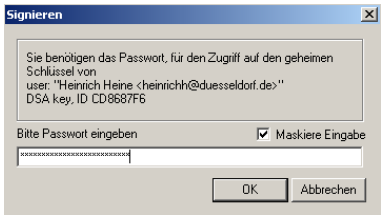
Sie können nun entscheiden ob Sie eine völlig unverschlüsselte, eine signierte oder eine komplett verschlüsselte Mail versenden wollen – je nachdem, wie wichtig und schutzbedürftig der Inhalt ist.

Dann öffnen Sie WinPT mit der rechten Maustaste aus der Windows-Taskbar und wählen im erscheinenden WinPT Menü *Zwischenablage→Signieren* aus. Anders als beim Verschlüsseln öffnet sich daraufhin ein Fenster mit Ihrem eigenen Schlüssel. Denn:

**Signieren können Sie nur mit Ihrem eigenen geheimen Schlüssel.**

Logisch, denn nur Ihr eigener Schlüssel bestätigt Ihre Identität. Der Korrespondenzpartner kann nun mit Ihrem öffentlichen Schlüssel, den er bereits hat oder sich besorgen kann, Ihre Identität überprüfen. Denn nur Ihr Geheimschlüssel passt ja zu Ihrem öffentlichen Schlüssel.

Klicken Sie also auf Ihren eigenen Schlüssel und bestätigen Sie mit [OK]. Im folgenden Fenster geben Sie Ihren geheimen Passwortsatz ein und bestätigen Sie wieder mit [OK]. Ein kurzer Hinweis erscheint sobald der Text signiert ist. Jetzt müssen Sie Ihren signierten Text nur noch in Ihr E-Mail- oder Textprogramm einfügen (Im WindowsMenü „Einfügen“ oder einfach Strg+V).



Ihre Nachricht ist nun am Anfang und Ende von einer Signatur eingerahmt:

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Werte Adele,

Wenn ich in deine Augen seh,
So schwindet all mein Leid und Weh;
Doch wenn ich küsse deinen Mund,
So werd ich ganz und gar gesund.

    Harry
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.3-cvs (MingW32)

iD8DBQFD36LVVyUTMs2Gh/YRAn2sAJ4wH2h8g+rFyxXQSsuYzZWzYMKTdgCeK0sK
CEL3//4INzHUNA/eqR3XMi0=
=tiQ5
-----END PGP SIGNATURE-----
```

Wenn Frau Adele diese E-Mail erhält, kann sie sicher sein,

1. dass die Nachricht von Herrn Heine stammt
2. dass sie nicht verändert wurde

Hätte zum Beispiel jemand das „gesund“ in dem obigen Beispiel zu „krank“ verändert, wäre die Signatur „gebrochen“, dass heißt, die E-Mail wäre mit dem Vermerk „Bad signature“ oder „Überprüfung fehlgeschlagen“ beim Empfänger versehen, sobald die Signatur überprüft wird.

## 10.2. Andere Gründe für eine gebrochene Signatur

Es gibt aber noch zwei weitere Gründe, die zu einem Bruch der Signatur führen können. Wenn Sie eine E-Mail mit dem Vermerk „Bad signature“ oder „Überprüfung fehlgeschlagen“ erhalten, ist das ein Warnsignal, muss aber nicht zwangsläufig bedeuten, dass Ihre E-Mail manipuliert wurde.

1. aufgrund der technischen Gegebenheiten ist es nicht auszuschließen, dass die E-Mail durch eine fehlerhafte Übertragung über das Internet verändert wurde.

2. das E-Mail-Programm des Absenders oder Empfängers kann falsch eingestellt sein. Wenn man eine signierte E-Mail verschickt, sollte man unbedingt darauf achten, dass im E-Mail-Programm alle Optionen ausgeschaltet sind, die E-Mail schon beim Versand verändern. Dazu zählt „HTML-Mails“ und „Word Wrap“.

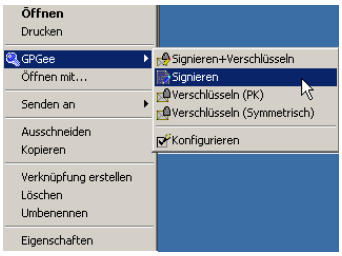
   „Word Wrap“ bezeichnet den Umbruch von Zeilen in der E-Mail. Beides verändert natürlich die E-Mail und „bricht“ die Signatur, obwohl niemand sie willentlich verändert hat. Bei Outlook Express beispielsweise muss diese Option unter „Extras / Optionen / Senden / NurText-Einstellungen / Textkodierung mit Keine“ aktiviert sein, wie es auch standardmäßig voreingestellt ist.

Häufig ist sind falsche Einstellungen am E-Mail-Programm der Grund für eine gebrochene Signatur. In beiden Fällen sollte man die E-Mail erneut anfordern.

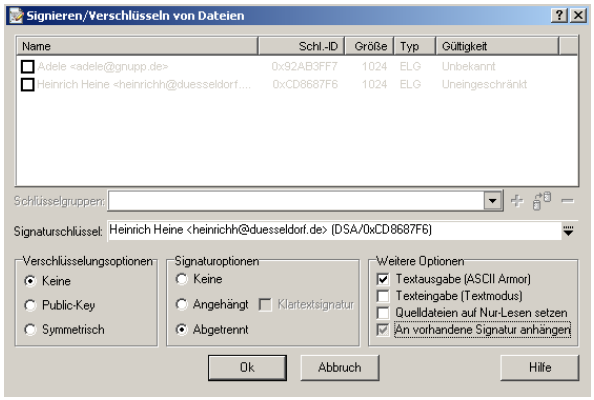
## 10.3. Dateien signieren

Nicht nur E-Mails, auch Dateien – z.B. ein PDF-Dokument — kann man signieren, bevor man sie per E-Mail verschickt oder per Diskette weitergibt. Auch dabei kommt es nicht vorrangig auf die Geheimhaltung, sondern auf die Unveränderheit der Datei an.

Diese Funktion können Sie bequem mit GPGee aus dem Kontextmenü des Explorers ausführen. Dazu öffnen Sie dessen Menü mit der rechten Maustaste:




Dort wählen Sie *signieren* aus, woraufhin das folgende Fenster erscheint:




Sie sehen in der Mitte eine Möglichkeit den Signaturschlüssel auszuwählen — nutzen Sie dies, falls Sie mit einem anderen als Ihrem Standardschlüssel signieren möchten.

Die drei Rahmen im unter Teil steuern die Signatur/Verschlüsselungs-Funktion; die Vorgaben sind in den meisten Fällen richtig. Die linke untere Box, steuert die Verschlüsselung. Da Sie lediglich signieren möchten ist hier „Keine“ ausgewählt.

In der mittleren Box können Sie die Art der Signatur wählen. Sie verwenden hier am besten eine „abgetrennte“ Signatur; dies bedeutet, dass die zu signierende Datei unverändert bleibt und eine zweite Datei mit der eigentlichen Signatur erzeugt wird. Um die Signatur später zu überprüfen sind dann beide Dateien notwendig.

In der rechten Box finden Sie noch weitere Optionen. „Textausgabe“ ist dort vorgegeben. Dies erzeugt eine abgetrennte Signaturdatei mit einem Dateinamen der auf „.asc“ endet und die direkt mit jedem Texteditor lesbar ist — sie würden dort den Buchstaben- und Ziffernsalat sehen, den Sie bereits kennen. Wenn diese Option nicht ausgewählt ist, so wird eine Datei mit der Endung „.sig“ erzeugt, die dann nicht direkt lesbar ist (binäre Datei). Was Sie hier benutzen ist eigentlich gleichgültig; Gpg4win kommt mit beiden Arten klar.

Zum Überprüfen der Unverändertheit und der Authentizität müssen die Original- und die signierte Datei im selben Verzeichnis liegen. Man öffnet die signierte Datei — also die mit der Endung „.sig“ oder „.asc“ — wieder aus dem Kontextmenü des Explorers mit *GPGee→Überprüfen/Entschlüsseln* .

Daraufhin erhalten Sie eine Ausgabe, ob die Signatur gültig ist — also die Datei nicht verändert wurde. Selbst wenn nur ein Zeichen hinzugefügt, gelöscht oder geändert wurde, wird die Signatur als ungültig angezeigt.

## 10.4. Verschlüsseln und signieren

Normalerweise verschlüsselt man eine Nachricht mit dem öffentlichen Schlüssel des Korrespondenzpartners, der ihn mit seinem privaten Schlüssel entschlüsselt.

Die umgekehrte Möglichkeit – man würde mit dem privaten Schlüssel verschlüsseln —, ist technisch nicht möglich und macht keinen Sinn, weil alle Welt den dazugehörigen öffentlichen Schlüssel kennt und die Nachricht damit entschlüsseln könnte.

Es gibt aber ein anderes Verfahren um mit Ihrem geheimen Schlüssel eine Datei zu erzeugen: Die Signatur, wie wir sie oben bereits beschrieben haben. Solch eine digitale Signatur bestätigt eindeutig die Urheberschaft – denn wenn jemand Ihren öffentlichen Schlüssel auf diese Datei (die Signatur) anwendet und die Ausgabe dieser Prüfung ist „gültig“, so kann diese Datei nur von Ihrem privaten Schlüssel kodiert worden sein. Und zu dem dürfen ja nur Sie selbst Zugang haben.

Wenn man ganz sicher gehen will, kann man beide Möglichkeiten kombinieren, also die E-Mail verschlüsseln und signieren:

1. Man signiert die Botschaft mit seinem eigenen geheimen Schlüssel. Damit ist die Urheberschaft nachweisbar.
2. dann verschlüsselt man den Text mit dem öffentlichen Schlüssel des Korrespondenzpartners.

Damit hat die Botschaft sozusagen zwei Briefumschläge:

1. einen Innenumschlag der mit einem Siegel verschlossen ist (die Signatur mit dem eigenen privaten Schlüssel) und
2. einen soliden äußeren Umschlag (die Verschlüsselung mit dem öffentlichen Schlüssel des Korrespondenzpartners).

Der Briefpartner öffnet die äußere, starke Hülle mit seinem eigenen geheimen Schlüssel. Hiermit ist die Geheimhaltung gewährleistet, denn nur dieser Schlüssel kann den Text dekodieren. Die innere, versiegelte Hülle öffnet er mit Ihrem öffentlichen Schlüssel und hat den Beweis Ihrer Urheberschaft, denn wenn Ihr öffentlicher Schlüssel passt, kann er nur mit Ihrem Geheimschlüssel kodiert worden sein.

Sehr trickreich und – wenn man ein wenig darüber nachdenkt – auch ganz einfach.

# 11. Dateianhänge verschlüsseln

Was tun, wenn Sie zusammen mit Ihrer E-Mail eine Datei versenden und diese ebenfalls verschlüsseln wollen? Die Verschlüsselung, wie wir sie in Kapitel 9 von „Gpg4win für Einsteiger“ erklärt haben, erfasst nur den Text der E-Mail, nicht aber eine gleichzeitig versandte, angehängte Datei.

Ganz einfach: Sie verschlüsseln den Anhang getrennt und hängen ihn dann in verschlüsseltem Zustand an die E-Mail an.

Und zwar so:

Klicken Sie die Datei mit der rechten Maustaste an und wählen Sie aus dem Menü *GPGee→ Verschlüsseln (PK)*. Sie sehen daraufhin das Fenster welches Sie schon im Kapitel „Dateien signieren“ kennengelernt haben.

Hier ist nun in der unteren linken Box „Public-Key“ markiert. Sie müssen jetzt im oberen Fenster auswählen, an welche Empfänger sie verschlüsseln wollen, kreuzen Sie einfach die entsprechenden Schlüsseln an.

Möchten Sie diese Datei (und damit auch den Dateianhang) auch noch signieren, so können Sie dies in der mittleren unteren Box auswählen („Angehängt“).

Die Datei wird verschlüsselt und mit der Endung `.gpg` im gleichen Ordner abgelegt wie die Originaldatei. Nun kann die verschlüsselte Datei wie jede andere als Attachment an eine E-Mail angehängt werden.

Viele Mailprogramme unterstützten das PGP/MIME Format, welches automatisch die Mail samt Anhängen verschlüsselt — in diesem Fall sollte das hier beschrieben Verfahren nicht angewandt werden. Einige anderer Mailprogramme verfügen über eine Option die das oben beschrieben Verfahren automatisch durchführen.

# 12. Im- und Export eines geheimen Schlüssels

Im „Schnellstart“-Handbuch haben wir in den Kapiteln 5, 6 und 8 den Im- und Export eines öffentlichen Schlüssels besprochen. Wir haben Ihren eigenen öffentlichen Schlüssel exportiert, um ihn zu veröffentlichen, und wir haben den öffentlichen Schlüssel Ihres Korrespondenzpartners importiert und „am Schlüsselbund“ befestigt.

Dabei ging es stets um den öffentlichen Schlüssel. Es gibt aber auch hin und wieder die Notwendigkeit, einen geheimen Schlüssel zu im- oder exportieren. Wenn Sie zum Beispiel einen bereits vorhandenen PGP-Schlüssel mit Gpg4win weiterbenutzen wollen, müssen Sie ihn importieren. Oder wenn Sie Gpg4win von einem anderen Rechner aus benutzen wollen, muss ebenfalls zunächst der gesamte Schlüssel dorthin transferiert werden – der öffentliche und der private Schlüssel.

Wir gehen im folgenden von der zur Zeit aktuellen PGP-Version 7 aus, in allen anderen ist der Vorgang ähnlich.

Zunächst speichern Sie beiden PGP-Schlüsselteile ab. Dazu müssen Sie in „PGPkeys“ Ihren Schlüssel anklicken und „Keys / Export“ anwählen. Auf dem Dateiauswahldialog „Export Key to File“ sehen Sie unten links eine Checkbox „Include Private Keys“, den Sie anklicken und mit einem Häkchen versehen müssen. PGP speichert beide Schlüsselteile in eine Datei ab, die Sie entsprechend benennen, zum Beispiel `geheimer-key.asc`.

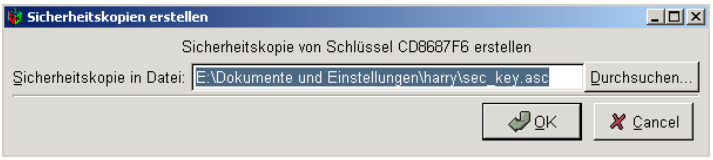
Öffnen Sie nun GPA oder WinPT und importieren sie einfach diese Datei. Es werden dann sowohl der geheime als auch der öffentliche Schlüssel importiert; sie sind dann sofort sichtbar. **Löschen Sie danach unbedingt die Datei `geheimer-key.asc` wieder und entfernen Sie diesen auch aus dem „Papierkorb“.** Damit haben Sie einen PGP-Schlüssel erfolgreich in Gpg4win importiert und können ihn dort genau wie einen normalen GnuPG-Schlüssel benutzen.

Es kann in einigen Fällen vorkommen, dass Sie einen importierten Schlüssel nicht direkt benutzen können. Dies äußert sich darin, dass Sie den richtigen Passwortsatz eingeben, dieser aber nicht akzeptiert wird. Das kommt daher, dass einige Versionen von PGP intern den IDEA Algorithmus verwenden. Dieser kann von GnuPG aus rechtlichen Gründen nicht unterstützt werden. Um das Problem zu beheben, ändern Sie in PGP einfach den Passwortsatz und exportieren/importieren Sie den Schlüssel erneut. Sollte dies auch nicht funktionieren, so setzen Sie den Passwortsatz in PGP auf „leer“; d.h. auf keinen Schutz und exportieren/importieren Sie wieder — In diesem Fall müssen Sie unbedingt sicherstellen, sowohl die **Datei sicher zu löschen als auch in PGP und in Gpg4win danach wieder einen echten Passwortsatz zu setzen.**

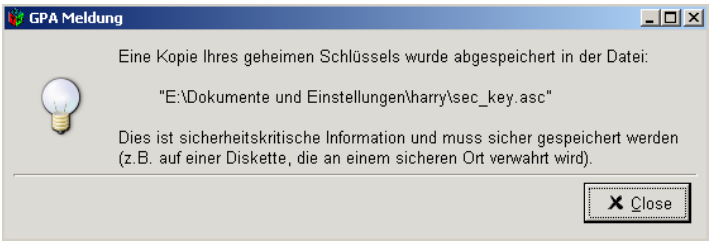
## 12.1. Export eines GnuPG-Schlüssels

Immer wenn Sie einen GnuPG-Schlüssel auf einen anderen Rechner transferieren oder auf einer anderen Festplattenpartition bzw. einer Sicherungsdiskette speichern wollen, müssen Sie mit WinPT oder GPA ein Backup erstellen. Dies entspricht dem Backup, welches Sie bei der Schlüsselerzeugung auch schon durchgeführt haben. Da Ihr Schlüssel inzwischen weitere Schlüsselunterschriften haben kann, sollte Sie es erneut durchführen.

Klicken Sie in der GPA-Schlüsselverwaltung den Schlüssel an, den Sie sichern wollen und wählen Sie dann den Menüpunkt *Schlüssel→Sicherheitskopie anlegen*.




Bestätigen Sie den Dateinamen oder wählen Sie einen anderen und GPA wird eine Sicherheitskopie bestehend aus dem geheimen und öffentlichen Schlüssel anlegen. Danach werden Sie noch daran erinnert, dass Sie diese Datei sehr sorgfältig zu handhaben ist:




Beim Import, also zum Beispiel auf einem anderen Rechner, importieren Sie einfach diese Sicherheitskopie in WinPT oder GPA. Gpg4win wird dann sowohl den geheimen als auch den öffentlichen Schlüssel aus dieser Datei importieren.

Damit haben Sie erfolgreich einen GnuPG-Schlüssel exportiert und wieder importiert.

# 13. Warum Gpg4win nicht zu knacken ist ...

... jedenfalls nicht mit heute bekannten Methoden und sofern die Implementierung der Programme frei von Fehlern ist.

In der Realität sind genau solche Fehler in den Programmen, Fehler im Betriebssystem oder nicht zuletzt Fehler in der Benutzung der letzte Weg um doch noch an die geheimen Informationen zu gelangen — Auch deshalb sollte Sie diese Handbücher bis hierhin gelesen haben.

In jedem Beispiel dieses Handbuchs haben Sie gesehen, dass zwischen dem geheimen und dem öffentlichen Schlüsselteil eine geheimnisvolle Verbindung besteht. Nur wenn beide zueinander passen, kann man Geheimbotschaften entschlüsseln.

Das Geheimnis dieser mathematischen Verbindung müssen Sie nicht unbedingt kennen – Gpg4win funktioniert für Sie auch so. Man kann diese komplexe mathematische Methode aber auch als Normalsterblicher und Nichtmathematiker verstehen. Sie müssen eigentlich nur einfache Additionen $(2 + 3)$ und Multiplikationen $(5 * 7)$ beherrschen. Allerdings in einer ganzen anderen Rechenmethode als der, die Sie im Alltag benutzen. Es gehört sowohl zur Sicherheitsphilosophie der Kryptographie wie auch zum Prinzip der Freien Software, dass es keine geheimnisvollen Methoden und Algorithmen gibt. Letztendlich versteht man auch erst dann wirklich, warum GnuPG (die eigentliche Maschinerie hinter Gpg4win) sicher ist.

Hier beginnt also sozusagen die Kür nach dem Pflichtteil:

# 14. GnuPG und das Geheimnis der großen Zahlen

## Kryptographie für Nicht-Mathematiker

Es ist schon versucht worden, den RSA Algorithmus, auf dem GnuPG basiert[4], zu „knacken“, also einen privaten Schlüssel zu berechnen, wenn man lediglich den öffentlichen Schlüssel kennt. Diese Berechnung ist aber noch nie für Schlüssellängen (1024 Bit und mehr), die in GnuPG verwendet werden, gelungen. Es ist theoretisch zwar möglich, aber praktisch undurchführbar da selbst bei genügend vorhandener Zeit (viele Jahre) und Abertausenden von vernetzten Rechnern niemals genügen Speicher zur Verfügung stehen wird, um den letzten Schritt dieser Berechnung durchführen zu können.

Es kann allerdings durchaus möglich sein, dass eines Tage eine geniale Idee die Mathematik revolutioniert und eine schnelle Lösung des mathematischen Problems, welches hinter RSA steckt, liefert. Dies wird aber wohl kaum von heute auf morgen geschehen. Das Bundesamt für Sicherheit in der Informationstechnik veröffentlicht von Zeit zu Zeit Prognosen und Einschätzungen, welche Schlüssellängen noch wieviele Jahre für absolute Geheimhaltung benutzt werden sollen. GnuPG überschreitet mit seinen Standardeinstellungen noch weit diese Mindestanforderungen. Wie im vorigen Kapitel schon angerissen, ist die Mathematik der mit Abstand sicherste Teil an der ganzen praktisch angewandten Kryptographie.

[4] Wir verwenden hier RSA als Beispiel da dieser einfacher zu verstehen ist als der Elgamal Algorithmus der als Voreinstellung von GnuPG benutzt wird.

Im Folgenden erfahren Sie, wie diese mathematische Methode funktioniert. Nicht in allen Einzelheiten – das würde den Rahmen dieser Anleitung bei weitem sprengen —, aber doch so, dass Sie bei etwas Mitrechnen selbst mathematisch korrekt ver- und entschlüsseln können und dabei das „Geheimnis der großen Zahlen“ entdecken.

Man kann diese komplexe mathematische Methode auch als Normalsterblicher und Nichtmathematiker verstehen. Sie müssen nur einfache Additionen und Multiplikationen beherrschen. Wie gesagt: hier beginnt der Kürteil, und bei der Kür geht es immer etwas mehr zur Sache als im Pflichtprogramm. Letztendlich versteht man dann aber, warum GnuPG sicher ist.

Eine Begriffsklärung vorneweg:

ein *Algorithmus* ist eine mathematische Prozedur zur Veränderung oder Transformation von Daten oder Informationen.

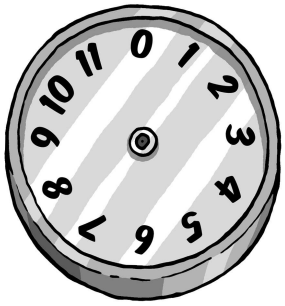
*Arithmetik* ist die Methode, nach der wir Zahlen addieren und multiplizieren.

Die Verschlüsselung mit GnuPG basiert auf dem sogenannten RSA-Algorithmus[5]. RSA steht für die Nachnamen von Ron Rivest, Ami Shamir und Ben Adleman, die diesen Algorithmus im Jahr 1978 entdeckt haben. Dieser Algorithmus verwendet einen Typ der Arithmetik, die Rechnen mit Restklassen oder „Modulo-Arithmetik“ heißt.

## 14.1. Das Rechnen mit Restklassen

Wenn man mit Restklassen rechnet, so bedeutet dies, dass man nur mit dem „Rest“ rechnet, der nach einer ganzzahligen Teilung durch eine bestimmte Zahl übrigbleibt. Diese Zahl, durch die geteilt wird, nennt man den „Modul“ oder die „Modulzahl“. Wenn wir beispielsweise mit dem Teiler oder der Modulzahl 5 rechnen, sagen wir auch, „wir rechnen modulo 5“.

Wie das Rechnen mit Restklassen — auch Modulo-Arithmetik oder Kongruenzrechnung genannt — funktioniert, kann man sich gut klarmachen, wenn man sich das Zifferblattes einer Uhr vorstellt:




Diese Uhr ist ein Beispiel für das Rechnen mit modulo 12 (der Teiler ist also 12) — eine Uhr mit einem normalen Zifferblatt, allerdings mit einer 0 anstelle der 12. Wir können damit Modulo-Arithmetik betreiben, indem wir einfach den gedachten Zeiger bewegen.

[5] RSA ist eigentlich optional, da aus Patentgründen der Elgamal Algorithmus, beruhend auf dem schwieriger zu erklärenden Problem des diskreten Logarithmus, als Standard verwendet wird.

Um beispielsweise $3 + 2$ zu rechnen, beginnen wir bei der Ziffer 2 und drehen den Zeiger um 3 Striche weiter (oder wir starten bei der 3 und drehen 2 Striche weiter, was natürlich auf dasselbe hinausläuft) Das Ergebnis ist 5.

Zählt man auf diese Weise $7 + 8$ zusammen, erhält man 3. Denn 3 ist der Rest, wenn man 15 (also $7 + 8$) durch 12 teilt. Um 5 mit 7 zu multiplizieren, beginnt man bei 0 und dreht 7 mal jeweils um 5 Striche weiter (oder auch bei 0 beginnend 5 mal um 7 Striche). In beiden Fällen bleibt der Zeiger bei 11 stehen. Denn 11 ist der Rest, wenn 35 (also $7 * 5$) durch 12 geteilt wird.

Beim Rechnen mit Restklassen addieren und teilen wir Zahlen also nach den normalen Regeln der Alltagsarithmetik, verwenden dabei jedoch immer nur den Rest nach der Teilung. Um anzuzeigen, dass wir nach den Regeln der Modulo-Arithmetik und nicht nach denen der üblichen Arithmetik rechnen, schreibt man den Modul (Sie wissen schon — den Teiler) dazu. Man sagt dann zum Beispiel „4 modulo 5“, schreibt aber kurz „4 mod 5“.

Bei Modulo-5 zum Beispiel hat man dann eine Uhr, auf deren Zifferblatt es nur die 0, 1, 2, 3 und 4 gibt. Also:

$$4 \bmod 5 + 3 \bmod 5 = 7 \bmod 5 = 2 \bmod 5$$

Anders ausgedrückt, ist in der Modulo-5 Arithmetik das Ergebnis aus 4 plus 3 gleich 2. Wir können also auch schreiben:

$$9 \bmod 5 + 7 \bmod 5 = 16 \bmod 5 = 1 \bmod 5$$

Wir sehen auch, dass es egal ist, in welcher Reihenfolge wir vorgehen, weil wir nämlich auch schreiben können:

$$9 \bmod 5 + 7 \bmod 5 = 4 \bmod 5 + 2 \bmod 5 = 6 \bmod 5 = 1 \bmod 5$$

Denn 4 ist dasselbe wie 9, und 2 dasselbe wie 7, da wir uns ja nur für den jeweiligen Rest nach der Teilung durch 5 interessieren. Daran wird deutlich, dass wir bei dieser Art der Arithmetik jederzeit 5 oder ein Vielfaches von 5, wie 10, 15 und so weiter nehmen können,und das Ergebnis stets dasselbe ist.

Das funktioniert auch beim Multiplizieren (Malnehmen).

Ein Beispiel:

$$4 \bmod 5 * 2 \bmod 5 = 8 \bmod 5 = 3 \bmod 5$$

Ebenso können wir schreiben:

$$9 \bmod 5 * 7 \bmod 5 = 63 \bmod 5 = 3 \bmod 5$$

da wir einfach 60, also $5 * 12$, abziehen können.

Man könnte aber auch schreiben:

$$9 \bmod 5 * 7 \bmod 5 = 4 \bmod 5 * 2 \bmod 5 = 8 \bmod 5 = 3 \bmod 5$$

denn 4 entspricht 9, und 2 entspricht 7, wenn wir nur den Rest nach Teilung durch 5 betrachten.

Widerum stellen wir fest, dass es egal ist, wenn wir das Vielfache von 5 einfach weglassen.

Da dadurch alles einfacher wird, machen wir das, bevor wir Zahlen addieren oder multiplizieren. Das bedeutet, dass wir uns lediglich um die Zahlen 0, 1, 2, 3 und 4 kümmern müssen, wenn wir mit der Modulo-5 Arithmetik rechnen. Denn wir können ja alles, was durch 5 teilbar ist, weglassen. Dazu noch drei Beispiele:

$$5 \bmod 11 * 3 \bmod 11 = 15 \bmod 11 = 4 \bmod 11$$

$$2 \bmod 7 * 4 \bmod 7 = 1 \bmod 7$$

$$13 \bmod 17 * 11 \bmod 17 = 7 \bmod 17$$

Das letzte Beispiel wird klar, wenn man bedenkt, dass in normaler Arithmetik gerechnet $13 * 11 = 143$ und $143 = 8 * 17 + 7$ ist.

## 14.2. RSA-Algorithmus und Rechnen mit Restklassen

Computer speichern Buchstaben als Zahlen. Alle Buchstaben und Symbole auf der Computertastatur werden in Wirklichkeit als Zahlen gespeichert, die zwischen zwischen 0 und 255 liegen.

Wir können also eine Nachricht auch in eine Zahlenfolge umwandeln. Nach welcher Methode (oder Algorithmus) dies geschieht, wird im nächsten Abschnitt beschrieben. Darin stellen wir Ihnen die Methode vor, nach der die Verschlüsselung mit GnuPG funktioniert: den RSA Algorithmus. Dieser Algorithmus wandelt eine Zahlenfolge (die ja eine Nachricht darstellen kann) so in eine andere Zahlenfolge um (Transformation), dass die Nachricht dabei verschlüsselt wird. Wenn man dabei nach dem richtigen Verfahren vorgeht, wird die Nachricht sicher kodiert und kann nur noch vom rechtmäßigen Empfänger dekodiert werden. Das sind die Grundlagen des RSA Algorithmus:

Sie selbst haben bei der Installation von Gpg4win während der Eingabe Ihres Passwortsatzes zwei große Primzahlen erzeugt, ohne es zu bemerken (dieser werden mit $p$ und $q$ bezeichnet). Nur Sie – oder in der Praxis Ihr Computer – kennen diese beiden Primzahlen, und Sie müssen für ihre Geheimhaltung sorgen.

Es werden daraus nun drei weitere Zahlen erzeugt:

**Die erste Zahl** ist das Ergebnis der Multiplikation der beiden Primzahlen, also ihr Produkt. Dieses Produkt wird als Modulus und dem Buchstaben $n$ bezeichnet. Dies ist der Modul mit dem wir später immer rechnen werden.

**Die zweite Zahl** ist der sogenannte öffentliche Exponent und eine Zahl an die bestimmte Anforderungen gestellt werden (teilerfremd zu $(p-1)(q-1)$); sie wird mit $e$ bezeichnet. Häufig wird hier 3, 41 oder 65537 benutzt.

**Die dritte Zahl** wird errechnet aus dem öffentlichem Exponent (der zweiten Zahl) und den beiden Primzahlen. Diese Zahl ist der geheime Exponent und wird mit $d$ bezeichnet. Die komplizierte Formel zur Berechnung lautet:

$$d = e^{-1} \bmod (p-1)(q-1)$$

Die erste und die zweite Zahl werden veröffentlicht – das ist Ihr öffentlicher Schlüssel. Beide werden dazu benutzt, Nachrichten zu verschlüsseln. Die dritte Zahl muss von Ihnen geheimgehalten werden – es ist Ihr geheimer Schlüssel. Die beiden Primzahlen werden danach nicht mehr benötigt.

Wenn eine verschlüsselte Nachricht empfangen wird, kann sie entschlüsselt werden mit Hilfe der ersten ($n$) und der dritten Zahl ($d$). Nur der Empfänger kennt beide Schlüsselteile – seinen öffentlichen und seinen geheimen Schlüssel. Der Rest der Welt kennt nur den öffentlichen Schlüssel ($n$ und $e$).

Die Trick des RSA Algorithmus liegt nun darin, dass es unmöglich ist, aus dem öffentlichen Schlüsselteil ($n$ und $e$) den geheimen Schlüsselteil ($d$) zu errechnen und damit die Botschaft zu entschlüsseln — denn: Nur wer im Besitz von $d$ ist, kann die Botschaft entschlüsseln.

## 14.3. RSA Verschlüsselung mit kleinen Zahlen

Wir verwenden hier erst einmal kleine Zahlen, um deutlich zu machen, wie die Methode funktioniert. In der Praxis verwendet man jedoch viel größere Primzahlen, die aus zig Ziffern bestehen.

Nehmen wir die Primzahlen 7 und 11. Damit verschlüsseln wir Zahlen – oder Buchstaben, was für den Computer dasselbe ist — nach dem RSA Algorithmus.

Und zwar erzeugen wir zunächst den öffentlichen Schlüssel

**Die erste Zahl** ist 77, nämlich das Ergebnis der Multiplikation der beiden Primzahlen, 7 und 11. 77 dient uns im weiteren Verlauf als Modulus zur Ver- und Entschlüsselung.

**Die zweite Zahl** ist der öffentliche Exponent. Wir wählen hier 13.

**Die dritte Zahl** ist der geheime Schlüssel. Sie wird in einem komplizierten Verfahren errechnet, welches wir jetzt erklären:

zunächst ziehen wir von unseren Primzahlen 7 und 11 jeweils die Zahl 1 ab (also $7-1$ und $11-1$) und multiplizieren die beiden resultierenden Zahlen miteinander. In unserem Beispiel ergibt das 60: $(7-1)*(11-1) = 60$. 60 ist unsere Modulzahl für die weiterführende Berechnung des geheimen Schlüssels (sie ist aber nicht mit dem eigentlichen Modulus 77 zu verwechseln).

Wir suchen jetzt eine Zahl, die multipliziert mit dem öffentlichen Schlüssel die Zahl 1 ergibt, wenn man mit dem Modul 60 rechnet:

$$13 \bmod 60 * \ ? \ \bmod 60 = 1 \bmod 60$$

Die einzige Zahl, die diese Bedingung erfüllt, ist 37, denn

$$13 \bmod 60 * 37 \bmod 60 = 481 \bmod 60 = 1 \bmod 60$$

37 ist die einzige Zahl, die multipliziert mit 13 die Zahl 1 ergibt, wenn man mit dem Modul 60 rechnet.

### 14.3.1. Wir verschlüsseln mit dem öffentlichen Schlüssel eine Nachricht

Nun zerlegen wir die Nachricht in eine Folge von Zahlen zwischen 0 und 76, also 77 Zahlen, denn sowohl Verschlüsselung als auch Entschlüsselung verwenden den Modul 77 (das Produkt aus den Primzahlen 7 und 11).

Jede einzelne dieser Zahlen wird nun nach der Modulo-77 Arithmetik 13 mal mit sich selbst multipliziert. Sie erinnern sich: die 13 ist ja unser öffentlicher Schlüssel.

Nehmen wir ein Beispiel mit der Zahl 2: sie wird in die Zahl 30 umgewandelt, weil $2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 = 8192 = 30 \bmod 77$ sind.

Ein weiteres Beispiel: 75 wird in die Zahl 47 umgewandelt, denn 75 wird 13 mal mit sich selbst multipliziert und durch 77 geteilt, so dass der Rest 47 entsteht.

Wenn man eine solche Rechnung für alle Zahlen zwischen 0 und 76 durchführt und die Ergebnisse in eine Tabelle einsetzt, sieht diese so aus:

In der linken Spalte stehen die 10er-Stellen, in der oberen Zeile die 1er-Stellen.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 30 | 38 | 53 | 26 | 62 | 35 | 50 | 58 |
| 10 | 10 | 11 | 12 | 41 | 49 | 64 | 37 | 73 | 46 | 61 |
| 20 | 69 | 21 | 22 | 23 | 52 | 60 | 75 | 48 | 7 | 57 |
| 30 | 72 | 3 | 32 | 33 | 34 | 63 | 71 | 9 | 59 | 18 |
| 40 | 68 | 6 | 14 | 43 | 44 | 45 | 74 | 5 | 20 | 70 |
| 50 | 29 | 2 | 17 | 25 | 54 | 55 | 56 | 8 | 16 | 31 |
| 60 | 4 | 40 | 13 | 28 | 36 | 65 | 66 | 67 | 19 | 27 |
| 70 | 42 | 15 | 51 | 24 | 39 | 47 | 76 | | | |

### 14.3.2. Wir entschlüsseln eine Nachricht mit dem privaten Schlüssel

Um das Beispiel mit der 2 von oben umzukehren, also die Nachricht zu dekodieren, multiplizieren wir 30 (die umgewandelte 2) unter Verwendung der Modulzahl 77 37 mal mit sich selbst. Sie erinnern sich: 37 ist der geheime Schlüssel.

Diese wiederholte Multiplikation ergibt eine Zahl die 2 mod 77 ergibt. Das andere Beispiel: die Zahl 47 mod 77 wird zur Zahl 75 mod 77 dekodiert.

Tabelle 2 zeigt die genaue Zuordnung der 77 Zahlen zwischen 0 und 76.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 51 | 31 | 60 | 47 | 41 | 28 | 57 | 37 |
| 10 | 10 | 11 | 12 | 62 | 42 | 71 | 58 | 52 | 39 | 68 |
| 20 | 48 | 21 | 22 | 23 | 73 | 53 | 5 | 69 | 63 | 50 |
| 30 | 2 | 59 | 32 | 33 | 34 | 7 | 64 | 16 | 3 | 74 |
| 40 | 61 | 13 | 70 | 43 | 44 | 45 | 18 | 75 | 27 | 14 |
| 50 | 8 | 72 | 24 | 4 | 54 | 55 | 56 | 29 | 9 | 38 |
| 60 | 25 | 19 | 6 | 35 | 15 | 65 | 66 | 67 | 40 | 20 |
| 70 | 49 | 36 | 30 | 17 | 46 | 26 | 76 | | | |

Um eine Zahl mit Tabelle 2 zu transformieren, gehen wir nach der gleichen Methode vor wie bei Tabelle 1. Ein Beispiel: 60 wird transformiert in die Zahl in Zeile 60 und Spalte 0. Also wird 60 zu 25 transformiert.

Das überrascht nicht, denn wenn wir davon ausgehen, dass wir bei der Umwandlung von 25 mit Hilfe von Tabelle 1 als Ergebnis 60 erhalten, dann sollten wir auch bei der Transformation von 60 mit Hilfe von Tabelle 2 zum Ergebnis 25 gelangen. Dabei haben wir den öffentlichen Schlüssel, 13, zur Umwandlung bzw. Kodierung einer Zahl verwendet, und den geheimen Schlüssel 37, um sie zurückzuwandeln bzw. zu dekodieren. Sowohl für die Verschlüsselung als auch für die Entschlüsselung haben wir uns der Modulo-77 Arithmetik bedient.

### 14.3.3. Zusammenfassung

Wir haben...

- durch den Computer zwei zufällige Primzahlen erzeugen lassen;
- daraus das Produkt und den öffentlichen und den geheimen Subkey gebildet;
- gezeigt, wie man mit dem öffentlichen Schlüssel Nachrichten verschlüsselt;
- gezeigt, wie man mit dem geheimen Schlüssel Nachrichten entschlüsselt.

Diese beiden Primzahlen können so groß gewählt werden, dass es unmöglich ist, sie einzig aus dem öffentlich bekannt gemachten Produkt zu ermitteln. Das begründet die Sicherheit des RSA Algorithmus.

Wir haben gesehen, dass die Rechnerei sogar in diesem einfachen Beispiel recht kompliziert geworden ist. In diesem Fall hat die Person, die den Schlüssel öffentlich gemacht hat, die Zahlen 77 und 13 als öffentlichen Schlüssel bekanntgegeben. Damit kann jedermann dieser Person mit der oben beschriebenen Methode – wie im Beispiel der Tabelle 1 – eine verschlüsselte Zahl oder Zahlenfolge schicken. Der rechtmäßige Empfänger der verschlüsselten Zahlenfolge kann diese dann mit Hilfe der Zahl 77 und dem geheimen Schlüssel 37 dekodieren.

In diesem einfachen Beispiel ist die Verschlüsselung natürlich nicht sonderlich sicher. Es ist klar, dass 77 das Produkt aus 7 und 11 ist.

Folglich kann man den Code in diesem einfachen Beispiel leicht knacken. Der scharfsinnige Leser wird auch bemerkt haben, dass etliche Zahlen, zum Beispiel die Zahl 11 und ihr Vielfaches (also 22, 33 etc.) und die benachbarten Zahlen sich in sich selbst umwandeln.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 51 | 31 | 60 | 47 | 41 | 28 | 57 | 37 |
| 10 | 10 | 11 | 12 | 62 | 42 | 71 | 58 | 52 | 39 | 68 |
| 20 | 48 | 21 | 22 | 23 | 73 | 53 | 5 | 69 | 63 | 50 |
| 30 | 2 | 59 | 32 | 33 | 34 | 7 | 64 | 16 | 3 | 74 |
| 40 | 61 | 13 | 70 | 43 | 44 | 45 | 18 | 75 | 27 | 14 |
| 50 | 8 | 72 | 24 | 4 | 54 | 55 | 56 | 29 | 9 | 38 |
| 60 | 25 | 19 | 6 | 35 | 15 | 65 | 66 | 67 | 40 | 20 |
| 70 | 49 | 36 | 30 | 17 | 46 | 26 | 76 | | | |

Das erscheint als ein weiterer Schwachpunkt dieser Verschlüsselungsmethode: man könnte annehmen, dass die Sicherheit des Algorithmus dadurch beeinträchtigt würde. Doch stellen Sie sich nun vor, das Produkt zweier grosser Primzahlen, die auf absolut willkürliche Art und Weise gewählt werden, ergäbe

114,381,625,757,888,867,669,235,779,976,146,612,010,
218,296,721,242,362,562,561,842,935,706,935,245,733,
89,830,597,123,563,958,705,058,989,075,147,599,290,
026,879,543,54

Hier ist überhaupt nicht mehr ersichtlich, welche die beiden zugrunde liegenden Primzahlen sind. Folglich ist es sehr schwierig, aufgrund des öffentlichen Schlüssels den geheimen Schlüssel zu ermitteln. Selbst den schnellsten Computern der Welt würde es gewaltige Probleme bereiten, die beiden Primzahlen zu errechnen.

Man muss die Primzahlen also nur groß genug wählen, damit ihre Berechnung aus dem Produkt so lange dauert, dass alle bekannten Methoden daran in der Praxis scheitern. Außerdem nimmt der Anteil der Zahlen, die in sich selbst transformiert werden – wie wir sie oben in den Tabellen 1 und 2 gefunden haben — stetig ab, je größer die Primzahlen werden. Von Primzahlen in der Grössenordnung, die wir in der Praxis bei der Verschlüsselung verwenden, ist dieser Teil ist so klein, dass der RSA Algorithmus davon in keiner Weise beeinträchtigt wird.

Je größer die Primzahlen, desto sicherer die Verschlüsselung. Trotzdem kann ein normaler PC ohne weiteres das Produkt aus den beiden großem Primzahlen bilden. Kein Rechner der Welt dagegen kann aus diesem Produkt wieder die ursprünglichen Primzahlen herausrechnen – jedenfalls nicht in vertretbarer Zeit.

## 14.4. Die Darstellung mit verschiedenen Basiszahlen

Um zu verstehen, wie Nachrichten verschlüsselt werden, sollte man wissen, wie ein Computer Zahlen speichert und vor allem, wie sie in unterschiedlichen Zahlenbasen dargestellt werden können.

Dazu machen wir uns zunächst mit den Zahlenpotenzen vertraut.

Zwei hoch eins, das man als $2^1$ darstellt, ist gleich 2; zwei hoch drei, dargestellt als $2^3$, ist $2 * 2 * 2 = 8$; zwei hoch zehn, dargestellt als $2^{10}$, ist $2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 = 1024$.

Jede Zahl hoch 0 ist gleich 1, zum Beispiel $2^0 = 1$ und $5^0 = 1$. Verallgemeinert bedeutet dies, dass eine potenzierte Zahl so oft mit sich selbst multipliziert wird, wie es die Hochzahl (Potenz) angibt.

Das Konzept einer Zahlenbasis veranschaulicht zum Beispiel ein Kilometerzähler im Auto: das rechte Rad zählt nach jedem Kilometer eine Stelle weiter und zwar nach der vertrauten Abfolge der Zahlen

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 1, 2

und so weiter. Jedesmal, wenn das rechte Rad wieder 0 erreicht, zählt das Rad links davon eine Stelle hoch. Und jedesmal, wenn dieses zweite Rad die 0 erreicht, erhöht das Rad links davon um eins ... und so weiter.


Das rechte Rad zählt die einzelnen Kilometer. Wenn es eine 8 angezeigt, dann sind dies 8 Kilometer. Das Rad links davon zeigt jeweils die vollen zehn Kilometer an: eine 5 bedeutet 50 Kilometer. Dann folgen die Hunderter: steht dort 7, dann bedeutet dies 700 Kilometer.

Nach dem gleichen Prinzip stellen wir ja auch unsere normale Zahlen mit den Ziffern 0 bis 9 dar.

„578“, zum Beispiel, bedeutet $5 * 100 + 7 * 10 + 8$, und dies entspricht 578.

Hier haben wir die „5“ stellvertretend für fünfhundert, „7“ für siebzig und „8“ für acht. In diesem Fall ist die Basis 10, eine für uns vertraute Basis.

Also steht die rechte Ziffer für die Einer der betreffenden Zahl (d.h. sie wird mit 1 multipliziert), die Ziffer links davon steht für die Zehner (d.h. wird mit 10 multipliziert), die nächste Ziffer wiederum für die Hunderter (d.h. sie wird mit 100 multipliziert) und so weiter. Da wir Zahlen normalerweise zur Basis 10 darstellen, machen wir uns nicht die Mühe, die Basis extra anzugeben. Formal würde man dies bei der Zahl 55 mit der Schreibweise $55_{10}$ anzeigen, wobei die tiefgestellte Zahl die Basis anzeigt.

Wenn wir nicht zur Basis 10 darstellen, so müssen wir dies mit Hilfe einer solchen tiefgestellten Basiszahl anzeigen.

Angenommen, die Anzeige des Kilometerzählers hätte statt der Ziffern 0 bis 9 nur noch 0 bis 7. Das rechte Rädchen würde nach jedem Kilometer um eine Ziffer höher zählen, wobei die Zahlenfolge so aussehen würde:

$$0, 1, 2, 3, 4, 5, 6, 7, 0, 1, 2, \textit{undsoweiter}.$$

Unser Tacho zur Basis 8 stellt zum Beispiel folgende Zahl dar:

356

Die 6 auf dem rechte Rädchen zählt einzelne Kilometer, also 6 Kilometer.
Die 5 auf dem Rädchen daneben für $5 * 8$, also 40 Kilometer.
Die 3 links steht für je 64 Kilometer pro Umdrehung, also hier $3 * 8 * 8$ Kilometer.

So rechnet man also mit Zahlen zur Basis 8. Ein Beispiel: 728 bedeutet $7 * 8 + 2$, und das ist gleich „58“. Bei dieser Art der Darstellung steht die „2“ aus der 72 für 2, aber die „7“ steht für $7 * 8$.

Größere Zahlen werden schrittweise genauso aufgebaut, so dass $453_8$ eigentlich $4*64+5*8+3$ bedeutet, was 299 ergibt.

Bei $453_8$ steht die „3“ für 3, die „5“ für $5 * 8$ und die „4“ für $4 * 64$, wobei sich die „64“ wiederum aus $8 * 8$ herleitet.

Im angeführten Beispiel werden die Ziffern, von rechts nach links gehend, mit aufsteigenden Potenzen von 8 multipliziert. Die rechte Ziffer wird mit 8 hoch 0 (das ist 1) multipliziert, die links daneben mit 8 hoch 1 (das ist 8), die nächste links davon mit 8 hoch 2 (das ist 64) und so weiter.
Wenn man Zahlen zur Basis 10 darstellt, gibt es keine höhere Ziffer als 9 (also 10 minus 1). Wir verfügen also über keine Ziffer, die 10 oder eine größere Zahl darstellt. Um 10 darzustellen, brauchen wir zwei Ziffern, mit denen wir dann die „10“ schreiben können.
Wir haben also nur die Ziffern 0 bis 9.

So ähnlich ist es, wenn wir mit der Basiszahl 8 rechnen: dann haben wir nur die Ziffern 0 bis 7. Wollen wir zu dieser Basis eine höhere Zahl als sieben darstellen, müssen wir wieder zwei Ziffern verwenden. Zum Beispiel „9“ schreibt man als $11_8$, „73“ schreibt man als $111_8$.

Computer speichern Zahlen als eine Folge von Nullen und Einsen. Man nennt dies Binärsystem oder Rechnen mit der Basiszahl 2, weil wir nur die Ziffern 0 und 1 verwenden. Stellen Sie sich vor, wir würden die Kilometer mit einem Tachometer zählen, auf dessen Rädchen sich nur zwei Ziffern befinden: 0 und 1. Die Zahl $10101_2$ zum Beispiel bedeutet im Binärsystem

$$1 * 16 + 0 * 8 + 1 * 4 + 0 * 2 + 1 = 21$$

.

In der Computerei verwendet man auch Gruppen von acht Binärziffern, das wohlbekannte Byte. Ein Byte kann Werte zwischen 0 - dargestellt als Byte $00000000_2$ — und 255 — dargestellt als Byte $11111111_2$ — annehmen. Ein Byte stellt also Zahlen zur Basis 256 dar.

Zwei weitere Beispiele:

$$10101010_2 = 170$$

und

$$00000101_2 = 5$$

.

Da der Computer die Buchstaben, Ziffern und Satzzeichen als Bytes speichert, schauen wir uns an, welche Rolle dabei die Darstellung zur Basis 256 spielt.

Nehmen wir die Silbe „un“. Das „u“ wird im Computer als 117 gespeichert und das „n“ als 110.

Diese Zahlenwerte sind für alle Computer standardisiert und werden ASCII-Code genannt. Um alle Zahlen und Symbole darstellen zu können, benötigen wir auf dem Computer die 256 Zahlen von 0 bis 255.

Wir können also die Silbe „un“ durch die Zahl $117 * 256 + 110$ darstellen.
Entsprechend würde man die Buchstabenfolge „und“ mit der Zahl $117*65536+110*256+100$ darstellen, denn das „d“ wird durch 100 repräsentiert.
Wir haben hier also Zahlen und Symbole, die auf der Computertastatur als normale Zahlen zur Basis 10 stehen, intern durch Zahlen zur Basis 256 repräsentiert.

Entsprechend können wir aus jeder Nachricht eine große Zahl machen. Aus einer langen Nachricht wird also eine gewaltig große Zahl. Und diese sehr große Zahl wollen wir nun nach dem RSA Algorithmus verschlüsseln.

Wir dürfen allerdings dabei die Zahl, zu der die Nachricht verschlüsselt wird, nicht größer werden lassen als das Produkt der Primzahlen (Modulus). Ansonsten bekommen wir Probleme, wie wir gleich noch sehen werden.

Da die folgende Prozedur mehrere Schritte umfaßt, fassen wir sie zunächst zusammen und verfolgen dann die Einzelschritte:

1. Die Nachricht *aba, cad, ada* wandeln wir — wie gesehen — in Zahlen um.
2. Diese Darstellung zur Basis 4 wandeln wir in eine Darstellung zur Basis 10 um, damit wir zur Verschlüsselung die Tabelle 1 benutzen können, in denen die Zahlen ja auch auf 10er-Basis dargestellt werden. Dabei entsteht eine kodierte Nachricht zur Basis 10.
3. Um die Kodierung im Vergleich zum „Klartext“ zu erkennen, rechnen wir die zur Basis 10 kodierte Nachricht auf die Basis 4 zurück und wandeln sie dann wieder in eine Buchstabensequenz.
4. So entsteht aus der Nachricht *aba, cad, ada* die verschlüsselte Nachricht *dbb, ddd, dac.*

Und nun ausführlich:

1. Die Nachricht *aba, cad, ada* wandeln wir — wie gesehen — in Zahlen um.

Angenommen, wir beschränken uns bei den Nachrichten auf die 4 Buchstaben a, b, c und d. In diesem — wirklich sehr einfachen — Beispiel können wir die vier Buchstaben durch die Zahlenwerte 0, 1, 2 und 3 darstellen, und haben dann

$$a = 0, b = 1, c = 2 \text{ und } d = 3$$

.

Wir wollen nun die Nachricht „abacadaca“ verschlüsseln. Wir kodieren diese Nachricht mit Hilfe der Primzahlen 7 und 11, mit dem öffentlichen Schlüssel 77 und 13 und dem dazugehörenden geheimen Schlüssel 37. Dieses Beispiel kennen wir bereits aus dem früheren Kapitel: wir haben damit die Tabellen 1 und 2 konstruiert.

2. Diese Darstellung zur Basis 4 wandeln wir in eine Darstellung zur Basis 10 um, damit wir zur Verschlüsselung die Tabelle 1 benutzen können, in denen die Zahlen ja auch auf 10er-Basis dargestellt werden.

Weil wir vier Buchstaben für die Nachricht verwenden, rechnen wir zur Basis 4. Für die Rechnung modulo 77 müssen wir die Nachricht in Stücke von je drei Zeichen Länge zerlegen, weil die größte dreiziffrige Zahl zur Basis 4 die $333_4$ ist. Zur Basis 10 hat diese Zahl den Wert 63.

Würden wir stattdessen die Nachricht in vier Zeichen lange Stücke zerlegen, würde die Zahl zu Basis 4 den Wert 76 übersteigen und es würden unerwünschte Doppeldeutigkeiten entstehen. Folglich würde die Nachricht in dreiziffrigen Stücken nun

$$aba, cad, aca$$

ergeben. Geben wir den Zeichen nun ihre Zahlenwerte und vergessen dabei nicht, dass die Stücke dreiziffrige Zahlen zur Basis 4 darstellen.

Da wir die Buchstaben durch die Zahlen a = 0, b = 1, c = 2, d = 3 darstellen, wird die Nachricht zu

$$010_4, 203_4, 020_4$$

.

Zur Basis 10 wird diese Nachricht durch die Zahlenfolge 4, 35, 8 dargestellt. Warum? Nehmen wir zum Beispiel das mittlere Stück $203_4$:

$$\begin{array}{lll} 3 * 4^0, & \text{also } 3 * 1, & \text{also } 3 \\ 0 * 4^1, & \text{also } 0 * 4, & \text{also } 0 \\ 2 * 4^2, & \text{also } 2 * 16, & \text{also } 32 \end{array}$$

3. Jetzt können wir zur Verschlüsselung die Tabelle 1 benutzen, die ja zur Basis 10 berechnet wurde. Diese Tabelle benutzen wir, weil wir mit dem schon bekannten Schlüsselpaar arbeiten wollen. Dabei entsteht eine kodierte Nachricht zur Basis 10.

Zum Verschlüsseln der Nachricht nehmen wir jetzt Tabelle 1 zur Hilfe. Die Nachricht wird nun zu der Zahlenfolge 53, 63, 50 (zur Basis 10).

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 30 | 38 | 53 | 26 | 62 | 35 | 50 | 58 |
| 10 | 10 | 11 | 12 | 41 | 49 | 64 | 37 | 73 | 46 | 61 |
| 20 | 69 | 21 | 22 | 23 | 52 | 60 | 75 | 48 | 7 | 57 |
| 30 | 72 | 3 | 32 | 33 | 34 | 63 | 71 | 9 | 59 | 18 |
| 40 | 68 | 6 | 14 | 43 | 44 | 45 | 74 | 5 | 20 | 70 |
| 50 | 29 | 2 | 17 | 25 | 54 | 55 | 56 | 8 | 16 | 31 |
| 60 | 4 | 40 | 13 | 28 | 36 | 65 | 66 | 67 | 19 | 27 |
| 70 | 42 | 15 | 51 | 24 | 39 | 47 | 76 | | | |

4. Wiederum zur Basis 4 konvertiert, entsteht die verschlüsselte Nachricht.

Wird sie nun wieder zur Basis 4 konvertiert, ergibt die Nachricht nun $311_4, 333_4, 302_4$. Konvertiert man diese zu einer Buchstabensequenz, erhält man dbb, ddd, dac, was sich nun erheblich von der ursprünglichen Nachricht unterscheidet.

Man kehrt nun also den Prozeß um und transformiert die Zahlenfolge 53, 63, 50 mit Tabelle 2 und erhält die Sequenz 4, 35, 8. Und das entspricht, als Zahlenfolge genau der ursprünglichen Nachricht.

Anhand der Tabellen 1 und 2 können wir ebensogut Nachrichten unter Verwendung des geheimen Schlüssels (d.h. erst Tabelle 2 benutzen) verschlüsseln, dann mit dem öffentlichen Schlüssel (d.h. Tabelle 1 als zweites benutzen) dekodieren und damit unsere ursprüngliche Zahl wieder herstellen. Das bedeutet – wie wir bereits im Handbuch „Gpg4win für Einsteiger“ gesehen haben —, dass der Inhaber des geheimen Schlüssels damit Nachrichten unter Verwendung des RSA Algorithmus verschlüsseln kann. Damit ist bewiesen, dass sie eindeutig nur von ihm stammen können.

`Fazit:`

Wie Sie gesehen haben, ist die ganze Angelegenheit zwar im Detail kompliziert, im Prinzip aber durchaus nachvollziehbar. Sie sollen schließlich nicht nur einer Methode einfach nur vertrauen, sondern – zumindest ansatzweise – ihre Funktionsweise durchschauen. Sehr viele tiefergehende Details sind leicht in anderen Büchern (z.B. R. Wobst, „Abenteuer Kryptologie“) oder im Internet zu finden.

**Immerhin wissen Sie nun:** wenn jemand sich an Ihren verschlüsselten E-Mails zu schaffen macht, ist er durchaus so lange damit beschäftigt, dass er dann keine Lust mehr haben kann sie auch noch zu lesen...

# A. History

- „GnuPP für Durchblicker“, Auflage März 2002,
  Autoren: Manfred J. Heinze, TextLab text+media
  Beratung: Lutz Zolondz, G-N-U GmbH
  Illustrationen: Karl Bihlmeier, Bihlmeier & Kramer GbR
  Layout: Isabel Kramer, Bihlmeier & Kramer GbR
  Fachtext: Dr. Francis Wray, e-mediate Ltd.
  Redaktion: Ute Bahn, TextLab text+media
  Herausgegeben vom Bundesministerium für Wirtschaft und Technologie.
  Verfügbar unter `http://www.gnupp.de/pdf/durchblicker.pdf`.
- Revidierte nicht-veröffentlichte Version von TextLab text+media.
- „Gpg4win für Durchblicker“, Dezember 2005
  Autoren: Werner Koch, g10 Code GmbH
  Herausgegeben durch das Gpg4win Projekt.

# B. GNU Free Documentation License

Version 1.2, November 2002



51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA



### Preamble

The purpose of this License is to make a manual, textbook, or other functional and useful document “free” in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of “copyleft”, which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

# 1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The **"Document"**, below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as **"you"**. You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A **"Modified Version"** of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A **"Secondary Section"** is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The **"Invariant Sections"** are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The **"Cover Texts"** are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A **"Transparent"** copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called **"Opaque"**.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The **"Title Page"** means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section **"Entitled XYZ"** means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as **"Acknowledgements"**, **"Dedications"**, **"Endorsements"**, or **"History"**.) To **"Preserve the Title"** of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

## 2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

## 3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

## 4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled “History”, Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled “History” in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the “History” section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled “Acknowledgements” or “Dedications”, Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled “Endorsements”. Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled “Endorsements” or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version’s license notice. These titles must be distinct from any other section titles.

You may add a section Entitled “Endorsements”, provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

## 5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

## 6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

## 7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

## 8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of

some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

## 9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

## 10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

## ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright ©YEAR YOUR NAME. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.